Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE DE FI DE CARRERA

TÍTOL DEL PFC: Implementació d'Algoritmes de Cerca a GRID Utilitzant GridSim

TITULACIÓ: Enginyeria de Telecomunicació (segon cicle)

AUTOR: Andreu Pere Isern Deyà

DIRECTOR: Roc Messeguer Pallarès

DATA: 8 d'abril de 2008

Títol: Implementació d'Algoritmes de Cerca a GRID Utilitzant GridSim

Autor: Andreu Pere Isern Deyà

Director: Roc Messeguer Pallarès

Data: 8 d'abril de 2008

Resum

Un GRID és una xarxa distribuïda que permet compartir recursos de còmput i d'emmagatzematge a través d'una xarxa de comunicacions, com per exemple Internet. Suposen una alternativa a la computació distribuïda basada en supercomputadors, i aporten, entre altres característiques, una disminució dels costs de desplegament i de manteniment, així com un increment de l'escalabilitat i la possibilitat d'usar recursos remots infrautilitzats.

Actualment, l'arquitectura GRID està en procés de maduració i estandardització, a més de ser objecte de múltiples investigacions per millorar les seves prestacions. Un d'aquests aspectes en estudi, és el desenvolupament de nous algoritmes de cerca de recursos sobre el GRID. En el present Projecte Final de Carrera es pretèn implementar dos algoritmes de cerca utilitzant el simulador GridSim, per a posteriorment evaluar les seves propietats i característiques.

La implementació dels algoritmes es duu a terme mitjançant el desenvolupament d'una aplicació que s'articula entorn del simulador GridSim i que intenta amagar les seves complicacions, de manera que es genera un envoltori pensat per a que posteriorment pugui ser usat o extès per altres projectes o fins i tot per a desplegar-lo sobre un GRID real.

Title: Implementing GRID Searching Algorithms Using GridSim

Author: Andreu Pere Isern Deyà

Director: Roc Messeguer Pallarès

Date: April, 8th 2008

Overview

A GRID is a distributed network that aims computacional and data sharing resources through a communications network , like Internet. These networks are an alternative against distributed computation based on supercomputers and provide many characteristics, for example, helps to decrease the deployment costs and maintenance, and provide an increase of scalability and the possibility to use underused remote resources.

Nowadays, the GRID architecture is in a maduration and standardization process, and around them many investigations are being in process to improve their characteristics. An important study is focused to develop new resource searching algorithms on GRID networks. In this Projecte Final de Carrera, aims to implement two GRID searching algorithms using the GridSim simulator, and then, evaluate their properties and characteristics.

Algorithm implementation is make developing and application using GridSim simulator and trying to avoid it's complications generating a wrapper around GridSim Toolkit. This wrapper may helps to develop another projects or even deploy the application into a real GRID network.

Índex

Índex de Taules

Índex de Figures

INTRODUCCIÓ

El present Projecte Final de Carrera té com a objectiu clau el d'implementar i evaluar dos algoritmes de cerca dins d'una arquitectura GRID. Actualment, les xarxes GRID, capaces de compartir distribuïdament recursos de computació o repositoris de dades, d'una forma molt similar a les xarxes P2P de compartició de fitxers actuals que coneixem, estan experimentant un increment en el seu ús, així com en el seu rendiment i àmbits d'aplicació. Aquest augment del seu ús, en bona part es deu als avantatges que reporten i el relatiu baix cost que suposa el seu desplegament i funcionament si es compara amb altres arquitectures com els supercomputadors, a més d'aprofitar-se en moltes ocasions de capacitat de còmput infrautilitzada.

Però resten nombrosos punts de l'arquitectura GRID sobre els quals és necessari més investigació i noves propostes que venguin a millorar les actuals. Un d'aquests camps de desenvolupament és el de crear i implementar algoritmes eficients de cerca de recursos a través de la xarxa distribuïda, que minimitzin el cost de cerca i maximitzin el seu rendiment.

És en aquest camp on es centra el present document, el qual implementa i evalua les característiques de dos algoritmes de cerca de recursos sobre GRID. Per a fer-ho, s'usen una sèrie d'eïnes de simulació que, juntament amb un acurat disseny d'una aplicació que en fa ús, generen una sèrie de dades estadístiques que seran analitzades i sobre les quals s'emetran una sèrie de conclusions.

El document està estructurat de manera que intenta simplificar la comprensió de tot l'entorn sobre el qual es basa el projecte, realitzant una clara separació de les diferents parts que el conformen. Així doncs, en el primer capítol es realitza una descripció de les motivacions i dels objectius que duen a desenvolupar el projecte. Seguidament, en el segon capítol es detallen els conceptes teòrics necessaris per a entendre el posterior desenvolupament de l'aplicació, com són les definicions de GRID, de les topologies d'aplicació sobre les quals es pot implentar i els mateixos algoritmes de cerca evaluats. Una vegada s'ha explicat la part teòrica, el tercer capítol és útil per a realitzar un repàs a les eïnes i llibreries que s'usen per a la implementació. El capítol 4, per la seva part, és el que forma el gruix del projecte. En ell es detalla el procés de disseny i desenvolupament de l'aplicació, així com l'ús que s'ha donat a cada una de les eïnes i llibreries anteriorment esmentades. A més, al final del capítol, es realitza la verificació de la correcta implementació dels algoritmes, mitjançant la comparació del resultat teòric d'un exemple de cerca front el resultat obtingut amb l'execució del mateix exemple sobre el simulador per ambdós algoritmes. Una vegada verificats els algoritmes, es passa al capítol 5, on s'exposen les simulacions realitzades, amb les dades estadístiques obtingudes i el seu anàlisi, per a finalment emetre una sèrie de conclusions i hipòtesis. Per últim, en el capítol 6 es detallen les conclusions finals i globals de tot el projecte. A més, el document integra una sèrie d'annexes amb informacions complementàries importants, com diagrames UML i de fluxe de l'aplicació, taules de dades estadístiques, anàlisi extens sobre les simulacions amb múltiples gràfiques i formats de fitxers d'entrada i sortida de l'aplicació.

CAPÍTOL 1. DESCRIPCIÓ DEL PROJECTE

1.1. Raó i Oportunitat del Projecte

La potència i capacitat de càlcul requerida per abordar l'execució de molts tipus de projectes és cada vegada més elevada i crítica. És de tots conegut l'existència de superordinadors formats per milers d'unitats de processament (CPU) i milers de GB de memòria RAM que treballen conjuntament i en paral·lel per tal de processar operacions complicades i simulacions costoses de processos físics, investigacions mèdiques o fenòmens meteorològics. Aquests supercomputadors es caracteritzen per estar construïts per ser emplaçats en una única sala (de petita o grans dimensions depenent de la potència final desitjada del supercomputador), on tots els processadors i memòries estan unides mitjançant xarxes d'altíssima velocitat de fibra òptica però amb molt poca distància entre tots els elements.

Aquest sistemes informàtics tenen un cost molt elevat així com l'inconvenient de la limitació en l'escalabilitat i a més, la dificultat del manteniment que es requereix per tal de tenir apunt tot el sistema. Per contraposició, l'avantatge que presenten és el control absolut de l'arquitectura i el de conèixer exactament la capacitat de càlcul disponible, sent aquesta constant al llarg del temps.

Una altra aproximació a la supercomputació és la de distribuir la capacitat de càlcul al llarg d'una xarxa de comunicacions que fa de nexe d'unió entre les unitats de processament de dades, amb la particularitat que cada una d'aquestes estan separades geogràficament entre sí. L'aproximació distribuïda no era factible fins fa molts pocs anys, ja que les xarxes de comunicacions no tenien la suficient capacitat com per a transmetre la quantitat de dades necessària amb els requeriments de temps, rendiment i latència idonis. Actualment, les xarxes de comunicacions d'alta velocitat es segueixen desenvolupant a un ritme elevat i ara ja estan molt exteses a nivell mundial, i per tant, l'impediment que suposaven per a distribuir els recursos de còmput ja no és crític.

Per tant, sorgeixen diferents formes de computació distribuïda, entre elles el GRID. Amb poques paraules, un GRID exposa a qualsevol usuari, amb permisos per accedir-hi, recursos distribuïts, disposats per altres usuaris o organitzacions, sent els recursos tant de còmput com d'emmagatzematge. En aquest cas, un GRID se sembla molt a la filosofia de les actuals xarxa P2P, canviant la compartició d'arxius per la cesió de temps de computació de les màquines que formen part de la xarxa. Tal com succeeix amb les xarxes P2P, un element clau és la recerca o localització dels recursos emmagatzemats en el GRID. Aquest és un tema en desenvolupament i existeixen múltiples propostes d'algoritmes de cerca, cada una d'elles amb unes certes característiques i uns determinats camps d'aplicació depenent del tipus de xarxa sobre la qual s'implementa el GRID.

Així doncs, el present projecte intenta implementar i evaluar el funcionament de dos algoritmes de cerca de recursos dins d'una arquitectura GRID, quan el

GRID s'organitza en forma d'una topologia d'hipercub, la qual té unes determinades característiques que s'evaluaran en el present projecte. L'evaluació es realitzarà mitjançant l'ús d'un simulador, sobre el qual s'implementaran els dos algoritmes proposats, per a posteriorment emetre unes conclusions sobre els resultats obtinguts.

1.2. Objectius

L'objectiu clau del projecte és el d'implementar i evaluar dos algoritmes de cerca en GRID usant el Toolkit GridSim com a eïna de simulació. Per tal d'aconseguir-ho, es fixen una sèrie d'objectius importants, que són els següents:

- **Comprendre la topologia hipercub r-dimensional**. Primer de tot, ja que el que es pretén és implementar algoritmes de cerca sobre una topologia d'hipercub, és essencial entendre les característiques i la forma de generar aquest tipus de topologia.

- **Aprendre usar el Toolkit GridSim**. S'ha de comprendre l'ús que es pot donar a GridSim per tal de desenvolupar l'entorn de simulació.

- **Dissenyar una aplicació que faci ús de les característiques de GridSim**. Una vegada entès GridSim, s'han d'extendre les seves característiques per tal d'adaptar el paquet als requeriments del projecte.

- Desenvolupar un entorn on hi hagi una clara **separació entre capa de xarxa i capa d'aplicació**. S'ha de separar clarament el que és la capa de xarxa, o xarxa IP, i la xarxa P2P a nivell d'aplicació que implementa el GRID. D'aquesta forma, es poden evaluar els algoritmes de nivell d'aplicació en contraposició a la topologia de xarxa IP subjacent.

- **Extreure estadístiques de les simulacions realitzades**. Amb una bona base estadística recolectada al llarg de les simulacions es podran fer anàlisis sobre el comportament dels algoritmes implementats. S'hauran de guardar mètriques tant a nivell de xarxa IP com del nivell d'aplicació.

- **Generar topologies a nivell IP**. És molt útil trobar alguna forma de generar diferents topologies IP depenent de paràmetres d'entrada, a més de que siguin aproximacions realistes a les xarxes desplegades. Com que l'objectiu en sí no és el de dissenyar les topologies, s'haurà de buscar algun software o aplicació que realitzi aquesta tasca, i integrar-la dins de l'aplicació desenvolupada.

CAPÍTOL 2. CONCEPTES TEÒRICS

2.1. Introducció

En aquest capítol s'explicaran els fonaments teòrics sobre els quals es fonamenta el desenvolupament del PFC. Primerament es definirà el concepte GRID, explicant els components que el formen, la seva arquitectura i la classificació d'un GRID depenent d'algunes de les seves característiques. A més, es detallarà l'estat actual de la tecnologia, enumerant els diferents estàndars que la defineixen, així com exemples d'aplicació reals del GRID.

A continuació, una vegada clar el concepte de GRID, es passa a definir les diferents topologies d'aplicació disponibles sobre les quals es pot implementar un GRID. Més concretament, es detallarà la topologia hipercub, objectiu d'implementació en el projecte, sobre la que s'evaluaran dos algoritmes de cerca de recursos, que també seran explicats en aquest mateix capítol.

2.2. GRID

2.2.1. Definició

El GRID (veure **[1]**, **[2]** i **[6]**) integra xarxes de comunicacions, computació i informació per a proporcionar un plataforma virtual per a la computació i la gestió de dades de la mateixa forma que Internet integra recursos en un plataforma virtual d'informació. L'arquitectura GRID està transformant la ciència, els negocis, la salud i la societat, ja que augmenta de forma significativa la potència de càlcul coordinada i emmagatzemament distribuït usant la combinació de múltiples recursos individuals que per si mateixos no tenen una potència significativa.

Un usuari pot accedir al GRID per a usar un recurs (ordinador individual, arxiu de dades, etc.), o per tal d'usar múltiples recursos de forma agregada com una computadora virtual. El GRID permet als usuaris interaccionar i interactuar amb els recursos d'una manera uniformitzada, proporcionant una plataforma potent per a computació global i per a la gestió de dades.

Una altra forma de definir un GRID és comparar-lo amb la xarxa elèctrica (segons **[5]**). La **Taula 2.1** mostra una comparativa entre els dos GRIDs, mostrant també algunes de les característiques claus d'un GRID.

	Xarxa elèctrica	***GRID***
Transparència	Quan es connecta un aparell elèctric, no es sap d'on procedeix ni on s'ha generat l'energia. Sols es nota que endollant-lo funciona.	Quan es tramet una tasca a un GRID o s'hi emmagatzemen dades, no es coneix exactament quin recurs s'ha usat.
Infraestructura	La xarxa interconnecta els consumidors amb les plantes de generació d'energia a	El GRID connecta entre sí recursos computacionals com CPUs, espai d'emmagatzematge,

	través de centres de transformació, línies de transport, etc.	etc, i proporciona mecanismes per accedir-hi.
Omnipresència	L'energia elèctrica és present quasi per tots els llocs. Només és necessari un endoll i la connexió amb la xarxa.	Els recursos són accessibles des de qualsevol dispositiu connectat a Internet.
Utilitària	Es té connexió elèctrica allà on es demana. Es paga pel servei que es presta.	Si es demana accés als recursos d'un GRID es tenen, amb o sense pagament, depenent del GRID.

Taula 2.1. Comparació de la xarxa elèctrica front el GRID.

2.2.2. Paradigme de Computació Distribuïda

La computació distribuïda és un mètode de processament en el que diferents parts d'un programa informàtic s'executen simultàniament sobre dues o més computadores les quals es comuniquen entre si mitjançant una xarxa. La computació distribuïda està relacionada amb la computació en paral·lel, amb la diferència que aquesta última tècnica es refereix al processament en paral·lel d'una mateixa tasca sobre la mateixa màquina. Les dues tècniques requereixen la segmentació de la tasca a processar, amb la salvetat que la computació distribuïda pot presentar diferències importants en els nodes de còmput, com poden ser diferents sistemes d'arxius, diferents capacitats de processament, etc.

2.2.3. Components

Un GRID és la unió d'una sèrie de components que treballen conjuntament per a donar el servei requerit. Així doncs, la següent llista mostra els elements claus que formen un GRID (veure **[1]** i **[3]**):

- **Xarxa de comunicacions**. Per interconnectar els nodes que formen part del GRID ha d'existir una xarxa de comunicacions per sota que transporti els missatges i les peticions dels usuaris cap als nodes que les processen. Les característiques desitjables de la xarxa són el tenir una amplada de banda suficient, índexs baixos de latència i innexistència de colls d'ampolla en la xarxa.

- **Recursos**. Són els elements que exposa el GRID per tal que els seus usuaris hi puguin accedir i fer-ne ús. Els recursos poden ser de tipus computacional o d'emmagatzematge de dades.

- **Usuaris**. Els usuaris són els clients de la xarxa, és a dir, els actors que demanen accés als recursos i esperen els resultats proporcionats pels mateixos.

- **Aplicació software**. Es tracta de l'aplicació específica que s'encarrega d'executar les tasques del GRID requerides pels usuaris sobre els

recursos disponibles. Per exemple, el tractament computacional d'unes certes dades.

- **Pol·lítiques d'accés**. Cada GRID defineix els seus propis criteris d'accés i ús dels seus recursos, dotant a la xarxa d'un cert nivell de seguretat i autenticació.

2.2.4. Característiques

Un GRID generalment té una sèrie de característiques o requisits a complir, els més importants dels quals són (veure **[2]** i **[4]**):

- Col·laboració. Un GRID es caracteritza per la naturalesa distribuïda dels recursos, i és per això que la col·laboració entre múltiples dominis administratius i usuaris és bàsica.

- Agregació. Un GRID s'encarrega d'agregar múltiples recursos individuals distribuïts en un recurs virtual agregat amb majors capacitats de còmput, emmagatzematge, etc.

- Localització de recursos. Cada recurs presenta un identificador o un nom, el qual és usat per a ser localitzat dins de la xarxa.

- Virtualització. Un GRID ha de proporcionar una sèrie d'interfícies per tal d'amagar la complexitat de la xarxa subjacent i dels recursos que agrega. Aquesta característica és coneguda com a virtualització, la qual també proporciona una capa d'abstracció entre els usuaris i els recursos.

- Orientat a serveis. Una arquitectura GRID proporciona serveis seguint la filosofia d'arquitectura orientada a serveis, on aquests es presenten com l'element central.

- Heterogeneïtat. Un GRID està composat per una sèrie de recursos computacionals heterogenis, cada un dels quals pot presentar varietats importants de hardware i software amb diferents característiques de latència o rendiment.

- Control descentralitzat. El control del GRID està distribuït a través de diferents entitats, ja que en sí mateix el GRID el formen diferents organitzacions.

- Estandardització i interoperabilitat. Un GRID ha d'exposar interfícies estàndards pels serveis que necessiten relacionar-se amb altres per tal de crear una infraestructura tal que satisfaci els requeriments dels usuaris i que les tasques d'aquests puguin ser executades eficientment.

- Accés transparent. Un GRID ha de proporcionar un accés transparent als usuaris que fan ús dels seus recursos, sense que aquests hagin de conèixer la topologia o arquitectura amb la que es basa el GRID.

- Escalabilitat. L'arquitectura d'un GRID i la seva topologia ha de ser escalable tant en recursos, nombre d'usuaris i dades, de manera que el seu rendiment no es vegi afectat amb l'augment d'aquests paràmetres.

- Reconfigurable. Un GRID ha de ser reconfigurable dinàmicament, de manera que nous recursos s'agreguin i passin a estar disponibles a la xarxa automàticament. De la mateixa manera, si un node o un enllaç del GRID, per la raó que sigui, cau i passa a un estat d'indisponibilitat, la xarxa ha de ser capaç de reconfigurar-se amb l'absència del node o enllaç que ha fallat.

- Seguretat. L'accés segur als recursos que exposa el GRID és una de les característiques clau de l'arquitectura. S'han de proveir mecanismes d'autenticació i verificació d'usuaris i aplicacions, i aplicar polítiques d'accés als recursos.

2.2.5. Tipus

Els GRIDs normalment es pensa que estan relacionats només amb aplicacions que requereixen alt rendiment i grans recursos computacionals. Però realment, donada la gran flexibilitat de les arquitectures GRID, n'existeixen de molts tipus, que van des de GRIDs computacionals d'alt rendiment, a xarxes entre dispositius domèstics, passant per GRIDs de dades. A continuació es detallen una sèrie de classificacions de tipus de GRIDs (veure **[1]**).

Depenent de l'extensió del GRID:

- **IntraGRID**. És tracta d'un GRID dins d'una mateixa organització i amb una extensió geogràfica limitada. Per exemple, un GRID dins d'un campus universitari.

- **InterGRID**. Es pot definir com la unió de diferents intraGRIDs d'una mateixa organització, que estan separats geogràficament, i que tenen un accés limitat als seus recursos. Per exemple, un GRID de les universitats espanyoles, on hi té accés el personal investigador de les mateixes.

- **ExtraGRID**. És la màxima expresió d'un GRID d'extensió mundial (**Fig. 2.1**), amb enorme distribució dels recursos, nodes i usuaris que el formen. Un exemple seria SETI@home (veure 2.2.7. i 8.3.).

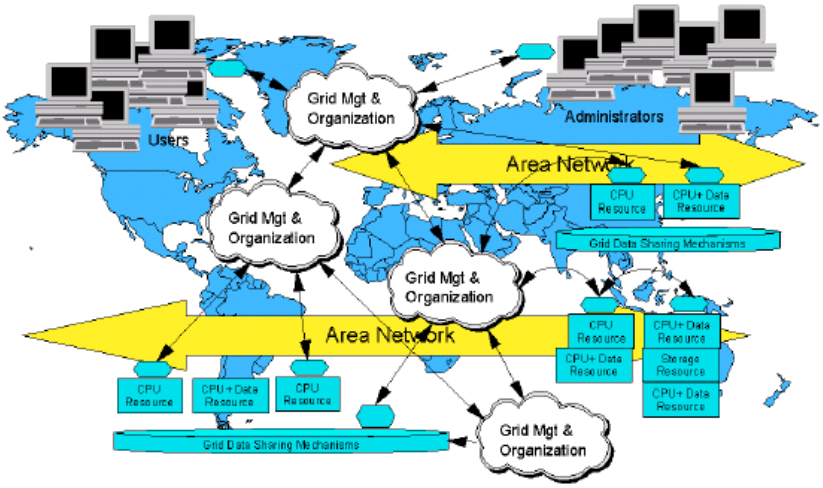


Fig. 2.1. Imatge esquemàtica d'un GRID.

Depenent dels recursos que exposa:

- **GRID computacional**. És el GRID que exposa recursos de càlcul, és a dir, capacitat de còmput.
- **GRID de dades**. És el GRID que proporciona recursos d'emmagatzematge de dades en comptes de capacitat de càlcul.
- **GRID de sensors**. És un tipus de GRID que està format per un conjunt de sensors, com per exemple meteorològics.

2.2.6. Arquitectura

L'arquitectura GRID està normalment definida en termes de capes, cada una de les quals duu a terme una determinada funció (veure **[9]**) (**Fig. 2.2**).

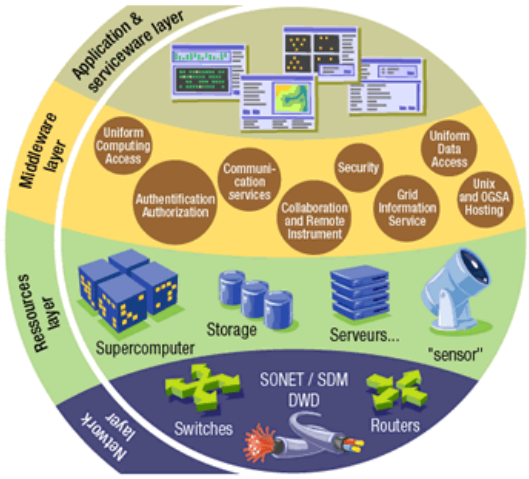


Fig. 2.2. Esquema de les capes que formen l'arquitectura de GRID.

Les capes altes estan centrades en l'usuari (*user-centric*), mentre que les baixes estan enfocades en encabir el hardware (*hardware-centric*): computadores i xarxes (veure **[1]** i **[5]**). Les capes que defineixen l'arquitectura són les següents (de nivell baix a alt):

Capa d'Infraestructura Física de la Xarxa

També anomenada *fabric*, proporciona els recursos que estan compartits per mitjà del GRID, com per exemple recursos computacionals, sistemes d'emmagatzematge, recursos de xarxes i sensors. Aquesta capa implementa les operacions locals i específiques del recurs (tant física com lògica).

Com a mínim, tot recurs ha d'implementar, d'una banda, mecanismes d'investigació que permetin descobrir la seva estructura, estat i capacitats, i per altra part, mecanismes de gestió dels recursos per a proporcionar cert control sobre la qualitat del servei ofert.

Capa Middleware. Connectivitat: Comunicacions i Seguretat

Aquesta capa defineix el nucli dels protocols de comunicació i autenticació requerits per a les transaccions realitzades dins el GRID.

- Els protocols de comunicació habiliten l'intercanvi de dades entre els recursos de la capa de xarxa. Inclouen el transport, l'enrutat i la identificació de les entitats que formen la xarxa.

- Els protocols d'autenticació, construïts sobre els serveis de comunicacions, són els encarregats de proporcionar eines criptogràfiques per a verificar la identitat dels recursos i dels usuaris.

Les solucions d'autenticació han de presentar les següents característiques:

- Single sing on. S'ha de permetre que un usuari realitzi una sola autenticació i que automàticament pugui accedir a múltiples recursos del GRID sense cap més intervenció per la seva part.

- Delegació. Un usuari ha poder dotar a una aplicació dels seus permisos d'autenticació per a que aquest programa pugui executar les seves funcionalitats sobre el GRID (delegació dels seus drets).

- Integració amb solucions de seguretat locals. Cada recurs pot presentar una forma particular de mecanisme d'autenticació (kerberos, UNIX, etc.), per tant, ha de ser capaç d'interaccionar amb diverses solucions de seguretat.

- Confiances basades en usuari. Si un usuari ha d'utilitzar recursos de diferents proveïdors, una vegada que estigui autenticat en cada un d'ells, els corresponents proveïdors no necessiten interactuar entre ells per a configurar l'entorn de seguretat.

Capa Middleware. Recursos: Compartir Recursos Individuals

La capa de recursos està construïda sobre la capa de connectivitat i autenticació per a definir protocols per a la negociació segura, inicialització, monitorització, control, *accounting* i pagament per a l'ús de recursos individuals. Dues classes de protocols es poden distingir en aquesta capa:

- Els protocols d'informació s'usen per a obtenir informació sobre l'estructura i l'estat del recurs, com per exemple, la seva configuració, la càrrega, etc.

- Els protocols de gestió s'utilitzen per a negociar l'accés al recurs compartit, especificant, els requeriments que es necessiten (reserva de recursos, qualitat de servei, etc.). També han de ser capaços de monitoritzar l'estat dels recursos d'una operació i controlar-la.

Capa Middleware. Serveis col·lectius: Coordinació de múltiples recursos

Mentre que la capa de recurs està centrada en les interaccions amb un recurs individual, la present capa conté protocols i serveis que no estan directament associats en cap recurs específic sinó que és global i captura les interaccions entre múltiples recursos. Pot implementar una àmplia varietat de polítiques de compartició sense que els recursos subjacents hagin de complir requeriments addicionals. Alguns exemples de serveis proporcionats per aquesta capa són:

- Serveis de directori, encarregats de proporcionar les propietats sobre els recursos als usuaris, així com l'existència dels mateixos.

- Planificació i assignació de recursos que permeten aplicar polítiques de planificació d'execució de tasques i l'assignació de les mateixes sobre determinats recursos.

- Serveis de monitorització i diagnòstic, capaços de monitoritzar els recursos per tal de detectar fallades, errors, sobrecàrregues, intrusions, etc.

- Serveis d'*accounting* i pagament, capaços de conèixer la utilització dels recursos per part dels usuaris, per qüestions de pagament o limitació de l'ús dels recursos.

Capa d'Aplicacions i Serveis

La capa de més alt nivell de l'arquitectura es correspon amb la capa d'aplicacions. Una aplicació utilitza els serveis proporcionats per les capes inferiors per a executar-se dins del GRID. Per exemple, una aplicació que ha d'analitzar una sèrie de dades ha de:

- Obtenir les credencials per accedir a les dades (protocols de recurs i connectivitat).

- Realitzar una petició al sistema d'informació per a determinar on estan situades les dades i quins recursos computacions poden ser usats (serveis col·lectius).

- Enviar peticions a la capa d'infraestructura física per extreure les dades, inicialitzar processos de computació i extreure els resultats (protocols de recurs i connectivitat).

- Monitoritzar el progrés de la tasca i notificar a l'usuari si el procés a conclòs satisfactòriament o si ha succeït alguna fallada.

2.2.7. Estat Actual i Futur del GRID

Un GRID, tal com s'ha fet notar al llarg d'aquest capítol, té la intenció de coordinar fonts de recursos distribuïts units entre sí per mitjà d'una xarxa de comunicacions, per a treballar coordinadament i augmentar la potència de càlcul disponible per a certes aplicacions de forma més econòmica.

Actualment, el concepte GRID està en constant desenvolupament i molt sovint es modifiquen les propietats i característiques desitjables d'un GRID. És a dir, no és encara una tecnologia madura, suficientment extesa i estandaritzada, sinó que existeixen múltiples propostes i especificacions disponibles.

D'entre els estàndards proposats o adoptats en l'arquitectura GRID es poden destacar els que s'enumeren a 8.2.

No obstant, existeixen múltiples projectes d'investigació que fan ús de la potència de càlcul que ofereixen els GRIDS: projectes mèdics, astronòmics, matemàtics, etc.

Exemples importants de GRID en funcionament actualment, entre molts d'altres, són SETI@home, Folding@home o GIMPS. Tots ells es detallen a 8.3.

2.3. Topologies GRID

Els nodes que formen part d'un GRID es poden organitzar en forma de múltiples topologies d'aplicació que marquen la forma de comunicació entre tots els nodes que la formen. L'elecció d'una topologia o altra és clau en el disseny d'una xarxa overlay, ja que condiciona les propietats de la mateixa. Existeixen, entre altres, les següents topologies:

- **Anell** (**Fig. 2.3.(1)**). En aquesta topologia els nodes estan organitzats sobre un cercle tancat, i cada node està connectat sols amb dos veïns.

- **Hipercub** (**Fig. 2.3.(2)**). Un hipercub és una topologia geomètrica amb una sèrie de característiques que s'expliquen en l'apartat 2.4.

- **Arbre** (**Fig. 2.3.(3)**). En un arbre, els nodes estan organitzats en forma de branques que es van ramificant, on tots i cada un dels nodes, excepte el root, tenen un pare. Excepte el node final d'una branca, tots els nodes tenen almenys un descendent.

- **Malla** (**Fig. 2.3.(4)**). És una xarxa on cada node està connectat amb un o més de la resta de nodes. D'aquesta forma, és possible fer arribar missatges als nodes a través de diferents camins. Si la xarxa està completament connectada, és impossible que existeixi interrupció en el servei.

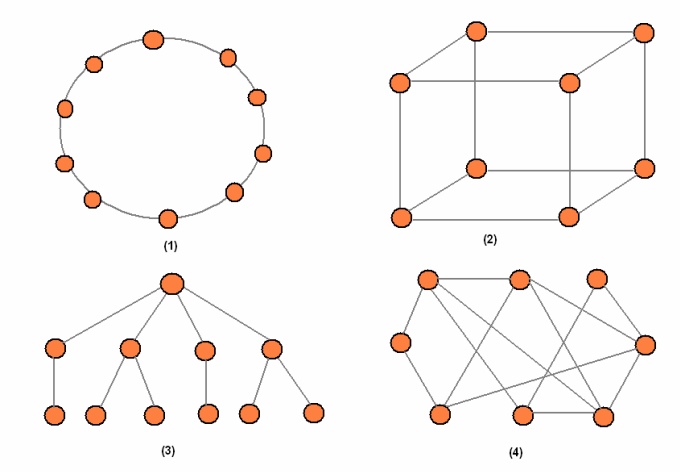


Fig. 2.3. Topologies de xarxa. **(1)** Anell. **(2)** Hipercub. **(3)** Arbre. **(4)** Malla.

En el present projecte, tal com s'ha especificat en els objectius, s'analitzarà i implementarà una xarxa hipercub per tal de desplegar-hi dos algoritmes de cerca de recursos. Aixi doncs, no és objectiu del projecte implementar ni evaluar altres topologies overlay diferents, restant com a objectiu d'un futur projecte si escau.

2.4. Xarxes hipercub r-dimensional. Característiques

Una de les xarxes més versàtils i usades és l'hipercub. Un hipercub H_r de dimensió r té $n = 2^r$ nodes, cada un dels quals té $r = logn$ vèrtexs. Es pot numerar cada un dels nodes del graf amb un string binari de longitud r. Dos nodes estan units mitjançant un enllaç si i només si els seus identificadors difereixen sols un bit.

En general, és més fàcil descriure un hipercub que dibuixar-lo. La **Fig. 2.4** mostra representacions d'hipercubs amb r=0,1,2,3,4. Observant detingudament la construcció d'un hipercub de dimensió r, es pot observar que tenen una estructura recursiva simple. Així doncs, un hipercub de dimensió $r+1$ es construeix a partir de dos hipercubs r dimensionals connectant els nodes corresponents. Anàlogament, ignorant el primer bit dels nodes de H_r, s'obtenen dos hipercubs H_{r-1} iguals.

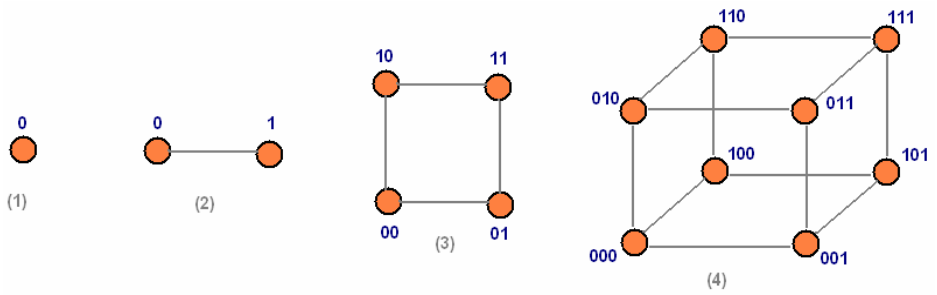


Fig. 2.4. Hipercubs. **(1)** r=0. **(2)** r=1. **(3)** r=2. **(4)** r=3.

Així doncs, l'hipercub H_r té les següents característiques clau:

- Diàmetre $r = logn$,
- número d'enllaços sortints des de cada node $r = logn$
- simetria
- estructura recursiva

En la **Fig. 2.4** cada un dels nodes està marcat amb un identificador binari, de longitud *r*, establert de manera que dos nodes veïns difereixin en sols 1 bit. Amb l'ús d'aquesta propietat, es pot crear una potent forma de routing.

2.4.1. Routing en un Hipercub

Tenint en compte que "*cada parell de nodes veïns difereixen exactament en un únic bit*", es pot generar el següent esquema de routing: per enviar dades des del node *x* al node *y*, es comparen els identificadors binaris de cada un dels nodes *x*, *y*, i en cada una de les etapes de routing, s'envien les dades al següent node variant únicament un dels bits de l'identificador. D'aquesta manera, es necessiten *logn* salts per arribar al destí.

Per exemple, per enviar dades des de 1011 fins a 0110, alguns possibles camins serien:

1011 → 0011 → 0111 → 0110
1011 → 1111 → 1110 → 0110

És interessant, per redundància en l'enrutat, el fet que existeixen múltiples camins per arribar a un determinat destí, tant és així, que si els identificadors de dos nodes difereixen en *k* bits, hi ha fins a *k!* camins diferents entre *x* i *y*.

En el present projecte s'analitza i s'usa una topologia hipercub r-dimensional per a muntar una xarxa GRID. S'analitzaran dos algoritmes de cerca i enrutat a través de la xarxa hipercub.

2.4.2. Algoritmes de cerca en Hipercub

Una de les funcionalitats claus que ha de tenir una xarxa distribuïda, i per tant, un GRID, és la de cerca de recursos, i és per això que s'han de dissenyar algoritmes per a desenvolupar aquesta tasca.

A continuació es detallaran dos algoritmes diferents de cerca de recursos dins d'un GRID: l'*algoritme P* i l'*algoritme H*.

2.4.3. Procés de cerca Algoritme P

El procés comença quan un client vol descobrir un servei GRID. El client realitza una petició contra un dels nodes d'informació del GRID, normalment el regional que té assignat per defecte, demanant el recurs desitjat. En primera instància, el denominat node inicial realitza una cerca a través dels recursos que conèix directament. Si no troba el recurs sol·licitat, el node s'encarrega de redirigir la petició del client als nodes veïns. En el pitjor dels casos, és a dir, quan no es troba el recurs sol·licitat en cap node, l'algoritme P realitza $log_2 N$ passes (la dimensió de l'hipercub) per arribar a tots els nodes (N) una sola vegada.

El fluxe d'execució de l'algoritme P es pot veure al diagrama de la **Fig. 2.5**.

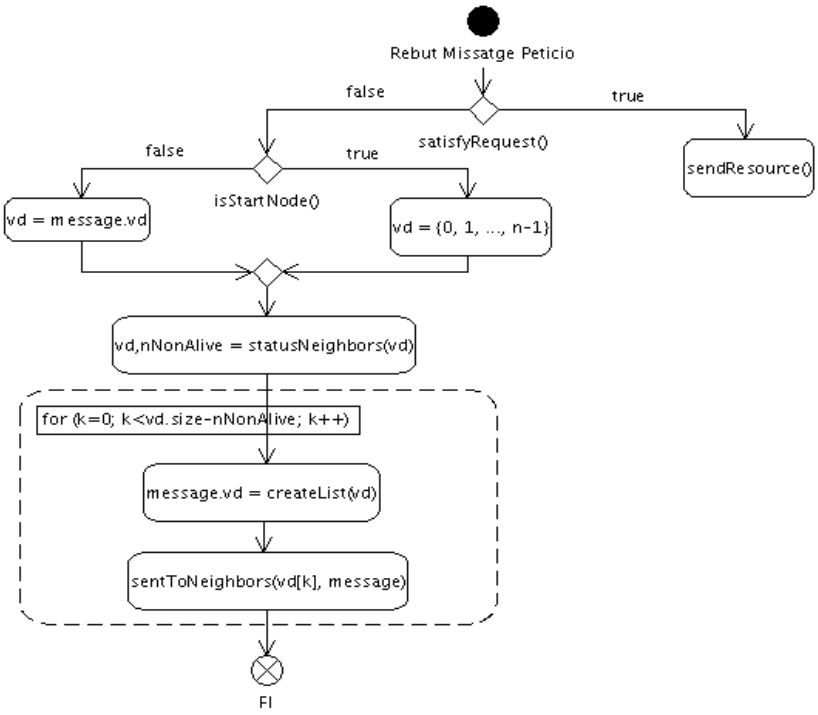


Fig. 2.5. Diagrama de fluxe de l'algoritme P.

Per tal de completar la informació, l'apartat 8.4.1. de l'annex 8.4. conté el pseudocodi i l'explicació extesa del funcionament de l'algoritme.

2.4.4. Procés de cerca Algoritme H

El procés de recerca de recursos, com en el cas de l'algoritme P, s'inicia quan es reb per part d'un usuari del GRID un missatge contenint una petició de recurs. El node receptor inicialment intentarà localitzar el recurs sol·licitat en local. Si no es troba, es reenvia la petició als nodes veïns. En el pitjor dels casos, és a dir, quan no es troba el recurs al llarg de tota la xarxa, el servei de cerca haurà de realitzar un total de log_2N+1 ($rdim + 1$) etapes per tal d'arribar a contactar una sola vegada tots els nodes. La **Fig. 2.6** mostra el diagrama de fluxe de l'execució de l'algoritme H quan es reb un missatge.

El pseudocodi de l'algoritme, així com l'explicació extesa de les passes que executa es poden consultar a l'apartat 8.4.2. dels annexes.

La propagació de les peticions d'aquest forma implica que l'efecte dels nodes no actius es veu reduït. Realitzant les modificacions oportunes sobre v_d, els nodes no actius propaguen la petició a un nombre menor de veïns que els actius. D'aquesta manera, l'algoritme intenta aïllar els nodes que estan en un estat de no activitat. Si cada node només té un veï no actiu, llavors tots els nodes actius poden ser consultats. Notar que un node pot esdevenir no actiu per estar massa carregat. En aquest cas, ja que no necessita propagar la petició rebuda, el seu estat no afecta el temps de resposta de la cerca. Per altra banda, els nodes que no poden ser accedits a causa de la no activitat d'altres, són accedits a través de la llista v_a a partir d'una altra branca de cerca.

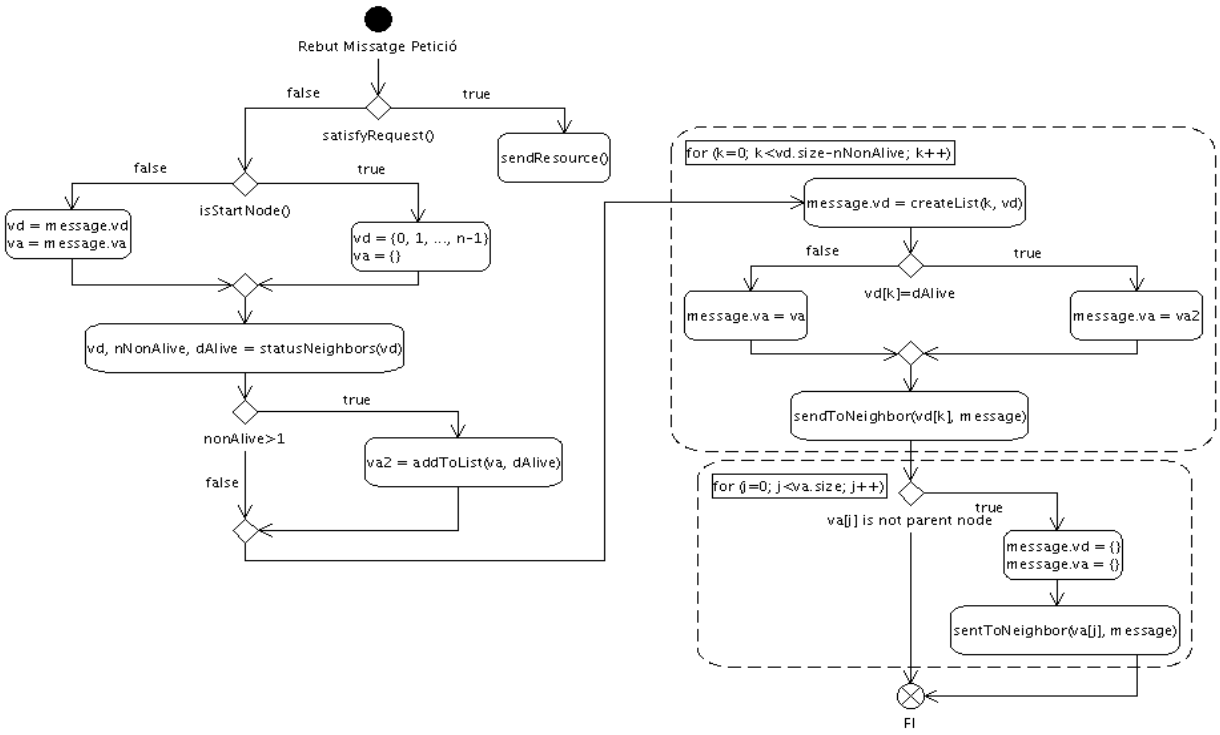


Fig. 2.6. Diagrama de fluxe algoritme H

CAPÍTOL 3. EÏNES DE DESENVOLUPAMENT

3.1. Entorn de Desenvolupament

En aquest capítol es detallaran les funcionalitats, la instal·lació i la configuració de les eïnes de desenvolupament usades, la majoria de les quals es basen principalment en Java, i per tant, és bàsic instal·lar i configurar correctament l'entorn. L'annex 8.5.1. explica les passes a seguir per a instal·lar Java.

Com aplicació IDE de desenvolupament s'ha usat Eclipse, ja que és un software molt extès i presenta un important suport de la comunitat open source. A més, s'hi poden instal·lar diversos plugins per tal d'augmentar les seves funcionalitats. Concretament, s'ha usat el plugin de Subversion, mitjançant el qual es gestiona el repositori de codi versionat (veure 8.5.5.).

Cal comentar que tot el desenvolupament s'ha realitzat sobre GNU/Linux, més concretament sobre sistemes Debian 4.0 (Lenny). No obstant, el fet d'usar Java fa que també sigui factible usar una versió Windows.

A continuació es detallen les llibreries més importants que s'han usat: GridSim, BRITE i Otter.

3.2. GridSim

GridSim (veure **[8]** i **[10]**) és un toolkit escrit íntegrament en Java que permet modelar i simular l'execució de processos i tasques en entorns de computació distribuïda. En aquest procés de simulació, es tenen en compte els usuaris, les aplicacions, els recursos i els planificadors de serveis per a dissenyar i evaluar el rendiment i l'eficiència d'un determinat algoritme de planificació, topologia de xarxa o política d'assignació de recursos. GridSim s'articula (**Fig. 3.7**) sobre SimJava (**[11]**), el qual és un paquet de simulació orientat a events. SimJava està programat amb Java i GridSim l'usa com a capa de nivell més baix, per a manejar els events i les intereccions entre les diferents entitats o objectes que defineix GridSim.

Tots els components de GridSim es comuniquen entre sí a través del pas de missatges definit per SimJava. La segona capa modela els elements de la infraestructura distribuïda, o recursos del GRID (clústers, repositoris d'emmagatzematge o enllaços de xarxa). La tercera i quarta capa s'encarreguen de modelar i simular els serveis específics de GRIDs computacionals o de dades, respectivament. Entre aquestes dues capes es defineixen serveis d'intercanvi d'informació entre elles, com informació de recursos disponibles. La cinquena capa conté components que permeten als usuaris implementar els seus propis planificadors, mentre que la sisena ajuda a definir escenaris i configuracions per a validar els algoritmes implementats.

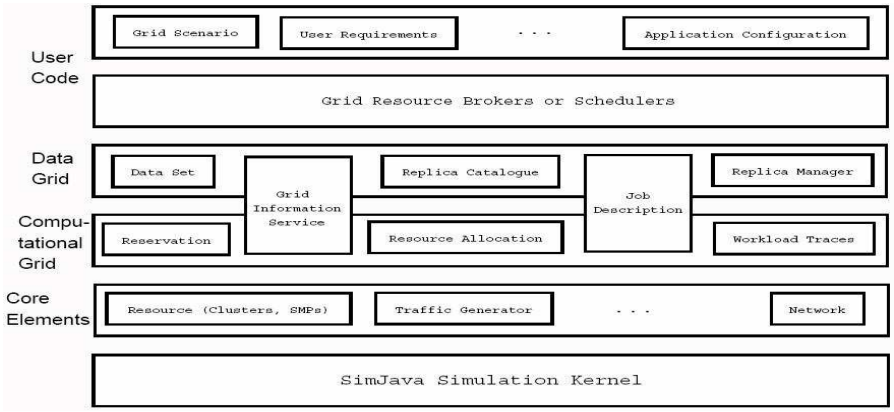


Fig. 3.7. Arquitectura de GridSim.

3.2.1. Funcionalitats

Les funcionalitats principals de l'eina són les següents:

- Incorpora caigudes dels recursos del GRID durant la seva execució.
- Definició de diferents polítiques de planificació i localització de recursos, i possibilitat de crear-ne de noves simplement estenent les pertinents classes.
- Incorpora extensions de xarxa per tal d'interconnectar recursos i altres entitats en una topologia de xarxa.
- Introdueix la funcionalitat de treballar amb tràfic de xarxa basat en distribucions probabilístiques. Aquesta funcionalitat és útil per a simular entorns GRID sobre xarxes públiques que poden presentar events de congestió.
- Creació d'algoritmes d'encaminament dins la xarxa GRID estenent les pertinents classes.

3.3. BRITE

BRITE (veure **[7]** i **[12]**) (Boston university Representative Internet Topology gEnerator) és una eïna de generació de topologies, que proporciona la capacitat de generar diversos escenaris mitjançant l'aplicació de models matemàtics, els quals pretenen simular topologies d'Internet, sobre les que realitzar estudis i simulacions.

Es tracta d'una aplicació d'escriptori, que es presenta amb dues versions, depenent del llenguatge de programació usat: Java i C++. No obstant, ambdues versions són iguals pel que fa a funcionament i característiques. Fer notar que les dues versions usen la mateixa interfície gràfica desenvolupada en Java. El que varia és el motor "background".

Cal remarcar que no s'intenta crear una o més topologies des de zero i analitzar la seva versemblança respecte de topologies reals, ja que no és objectiu del present projecte. El que es pretén és utilitzar les possibilitats que proporciona BRITE per a modelar una sèrie de topologies generades a partir de models matemàtics que suposen una aproximació a la realitat.

3.3.1. Funcionalitats

Les funcionalitats més importants de l'aplicació són les següents:

- Incorpora diversos models matemàtics per a la generació de topologies de xarxa, incloent posicionament de nodes i les característiques dels enllaços entre nodes (retard i amplada de banda disponible),

- Interfície gràfica per a la configuració del motor de generació de topologies, la qual modifica un fitxer de text pla que serveix d'entrada a l'aplicació,

- Diversos models de topologies: només routers, només AS (Sistemes Autònoms), model "top-down" i model "bottom-up",

- Interoperable amb eïnes externes: es proporciona la sortida del generador amb diversos formats de fitxer per a diferents simuladors i visualitzadors de topologia,

- Els models de generació i els fitxers de sortida són extensibles.

3.3.2. Models de Topologies

Flat router-level

En el model de nivell de router (**Fig. 3.8** (1)), es genera una matriu de routers, que es posicionen sobre un pla cartesià, per a posteriorment interconnectar-los mitjançant enllaços amb unes característiques propies de delay i amplada de banda. L'assignació i les característiques dels enllaços entre nodes es pot realitzar usant dos algoritmes diferents: Waxman **[7]** i Barabasi Albert **[7]**.

Flat AS-level

És un model (**Fig. 3.8** (2)) anàleg a l'anterior, amb la diferència que ara el que es posicionen són Sistemes Autònoms, amb la capacitat que cada un d'ells pot contenir una topologia interna. De la mateixa manera que en el model de nivell de router, es poden usar els dos algoritmes matemàtics esmentats.

Top-Bottom

Es tracta d'una topologia de xarxa amb estructura jeràrquica (**Fig. 3.8** (3)), on primer es crea el nivell de sistema autònom d'acord amb un dels dos models

disponibles, per a posteriorment generar topologies de nivell de router per a cada un dels sistemes autònoms definits.

Bottom-Up

Finalment el model Bottom-Up (**Fig. 3.8** (4)), genera una topologia on el que primer es genera és el nivell de router, per a posteriorment assignar a cada node AS un determinat número de routers.

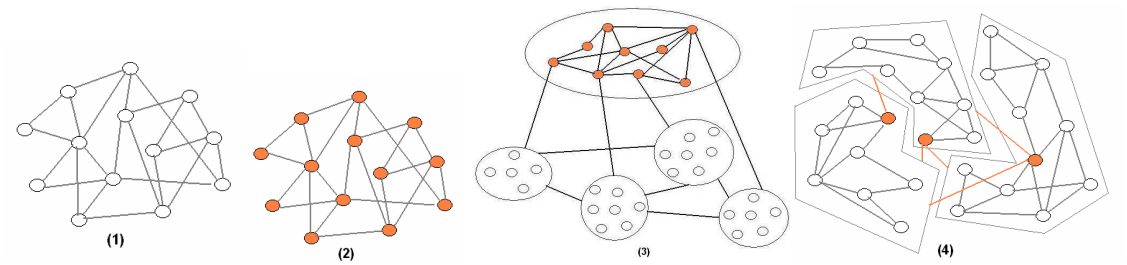


Fig. 3.8. Esquemes dels models de topologia que BRITE genera. **(1)** Flat router-level. **(2)** Flas AS-level. **(3)** Top-Bottom. **(4)** Bottom-Up.

3.4. Otter

Otter (**[13]**) és una aplicació Java desenvolupada per CAIDA per a la visualització de dades de xarxes que poden ser definides en forma de nodes, enllaços i camins. És capaç de mostrar representacions visuals de topologies de xarxa i de característiques pròpies de les mateixes com, tràfic cursat, retards entre nodes, distàncies geogràfiques, etc.

Actualment, Otter no es segueix desenvolupant i sols s'hi realitzen correccions molt esporàdiques per tal d'arreglar alguns errors.

3.4.1. Funcionalitats

La funcionalitat principal és la de mostrar una representació de la topologia de xarxa. A partir d'aquí, es defineixen una sèrie de funcionalitats claus:

- Entrada de les dades de la xarxa en un fitxer de text especificat a 8.11.

- Diferents visualitzacions de la mateixa topologia:
  - posicionament geogràfic
  - posicionament semigeogràfic
  - posicionament en forma de cercle
  - posicionament basat en el valor

- Capacitat per a pintar dades sobre la topologia usant una gama de colors. Per exemple, si es visualitza el retard entre dos nodes veïns, Otter coloreja cada un dels enllaços segons el valor de retard definit.

- Zoom. Permet augmentar o disminuir el zoom de visualització

CAPÍTOL 4. DISSENY I IMPLEMENTACIÓ

4.1. Introducció al Disseny Orientat a Objectes

L'objectiu clau del disseny de l'aplicació del present projecte és el d'utilitzar al màxim el concepte d'orientació a objectes de Java per tal d'aconseguir un desenvolupament òptim pel que fa a codi, i crear una estructura clara que faciliti la posterior extensió de l'aplicació i la inclusió de noves característiques sempre que sigui necessari. Per tant, s'usen característiques d'orientació a objectes, com l'herència, el polimorfisme, les classes abstractes, i patrons de disseny com el singleton, la factory o el command.

Al llarg d'aquest capítol, s'explicaran les passes més importants que s'han dut a terme en el disseny i en l'ús de l'orientació a objectes, fins arribar a l'arbre de classes UML definitiu (8.7.) amb totes les dependències necessàries, a més de presentar un diagrama de fluxe general (8.8.3.) per a tota l'aplicació.

4.2. Línia de Desenvolupament

Analitzant les característiques dels GRIDs i els elements que els formen, al llarg del desenvolupament s'esmentaran els objectes que formaran part de la xarxa i per tant, que participaran en les simulacions. Així doncs, aquest és un bon punt per a definir a alt nivell aquests objectes:

- **Router**. Com en un xarxa real, un router és l'element que s'encarrega de les tasques d'enrutat de paquets, és a dir, d'adreçar els paquets des d'un origen cap a un destí, identificats ambdós per una adreça única en tot l'espai de la xarxa.

- **Enllaç**. És un enllaç físic, que a la realitat pot ser de tipus òptic, elèctric o radio, que interconnecta entre sí dos elements de xarxa allunyats físicament.

- **GIS**. Un GIS (sigles de GRID Information Service) és l'element central d'un GRID, el qual concentra la intel·ligència del mateix. Així doncs, en l'entorn de simulació, un GIS és un node a nivell d'aplicació que s'encarrega de gestionar les peticions de cerca dels usuaris, i emmagatzema informació dels recursos que té adherits. A la realitat, un GIS normalment és exposat per una organització que cedeix els seus recursos de còmput o magatzem de dades al GRID.

- **Usuari**. Un usuari és l'actor que resideix a l'exterior del GRID i que s'hi connecta sempre que se li fa necessari adquirir recursos compartits a través del GRID.

- **Recurs**. És qualsevol element tant de còmput com d'emmagatzematge de dades disponible a través de la plataforma GRID, i que es deixa a disposició dels usuaris i es registre en el GIS corresponent.

- **Missatge**. Per tal d'obtenir comunicacions entre els elements del nivell d'aplicació del GRID, s'envien missatges, que són encaminats pels routers i transmessos a través dels enllaços.

Per una altra part, una vegada definits els elements clau que formen l'entorn GRID, és útil detallar el fluxe seguit per al desenvolupament de l'aplicació:

- Primer de tot, s'ha de realitzar un estudi del toolkit GridSim, per tal de comprovar les funcionalitats que proporciona i les característiques que disposa al programador.

- Una vegada es tenen clars els objectes a usar i que participaran directament en el desenvolupament, s'ha de dissenyar la forma d'usar-los dins de l'aplicació, integrant-los tots baix un sol concepte: el node GRID.

- Ja que l'entorn a simular és en el fons una xarxa de nodes interconnectats entre sí d'una certa manera, formant així una topologia a nivell de xarxa, s'han de trobar mecanismes per tal de generar-la i que proporcioni una plataforma de connexió de tots entre tots. Els objectes clau en aquest nivell són els routers i els enllaços entre ells.

- Una vegada que la topologia subjacent està generada, el que s'ha de resoldre és de quina manera generar la topologia overlay a nivell d'aplicació a partir de la xarxa que té per sota.

- Per a dissenyar la topologia d'aplicació, s'han de tenir clars els actors que participen en aquest nivell: GIS, usuaris i recursos.

- Tot seguit, ja que l'objectiu és implementar dos algoritmes de cerca de recursos a través de la xarxa implementada, s'ha de comprovar de quina manera implementar-los i com llançar els processos de cerca des dels usuaris. Els elements sobre els quals s'implementen els algoritmes són els GIS.

- Paral·lelament a tots els anteriors punts, s'ha de dissenyar la forma d'emmagatzemar les dades de topologia generades (tant d'aplicació com de xarxa), de forma que puguin ser fàcilment consultades al llarg de tota l'aplicació i en qualsevol moment que es desitgi. És a dir, que estiguin sempre disponibles i que a més, si cal, siguin modificables.

- A més d'emmagatzemar les dades de topologia, també s'han de guardar els resultats i les mètriques que siguin necessàries, generades per l'aplicació, referents als processos de cerca de recursos a través del GRID.

- Finalment, i com a complement visual, s'ha d'analitzar si existeix alguna forma de mostrar un esquema visual de la topologia completa generada.

4.3. Estudi del Toolkit GridSim

Primer de tot cal realitzar un estudi de GridSim, per evaluar el seu funcionament, les característiques aprofitables i la forma d'extendre'l. Per això, es molt útil consultar l'arbre de classes UML de GridSim, contingut a 8.6.

GridSim és essencialment una API a partir de la qual es creen scripts i aplicacions que fan ús de les seves funcionalitats. GridSim usa SimJava per executar-se i realitzar les simulacions.

En principi, pel que es pot evaluar amb els exemples que proporciona, GridSim té com a objectiu el de simular l'enviament de tasques al GRID (que anomena Gridlets) per a que siguin executats pels recursos distribuïts, tot seguint els planificadors i polítiques de servei que incorpora o extenent-ne de nous per tal d'evaluar-los. Per tant, no existeixen exemples ni documentació explicativa per a implementar algoritmes de cerca tal com es vol en el projecte. Finalment, després d'investigar el codi i la API, es determina que els objectes de xarxa que defineix (router, usuari, GIS i recurs) són extensibles, si bé les funcionalitats que proporcionen són molt limitades i cal ampliar-les.

A part dels nodes, és clau estudiar la forma de comunicació entre els nodes. GridSim defineix una classe que implementa un paquet de xarxa, així com una extensió d'aquest per a realitzar pings. Així doncs, sembla oportú partir d'aquesta base i crear un nou tipus de paquet per a transportar les dades d'aplicació entre els usuaris i els GIS.

La forma de guardar els objectes de xarxa creats i que seran usats en la posterior simulació és mitjançant un singleton. Per tant, només existeix un punt d'atac des d'on guarda i extreu aquests objectes i les seves propietats, que és la classe *GridSim*.

Una vegada compresos els exemples, evaluada l'API i havent conegut per sobre el funcionament, ja es pot iniciar el procés de disseny de l'aplicació.

4.4. Desenvolupar un Envoltori entorn a GridSim

L'objectiu clau del disseny de l'aplicació és el de crear un envoltori entorn de GridSim, que consegueixi amagar les complicacions del Toolkit, i a la vegada extendre les seves funcionalitats adaptant-les a les desitjades per a la implementació dels algoritmes de cerca. Aquest punt és molt interessant, ja que es pot aconseguir que substituïnt GridSim per un GRID real, l'aplicació funcioni correctament sols desenvolupant una capa de d'adaptació entre ells (**Fig. 4.9**).

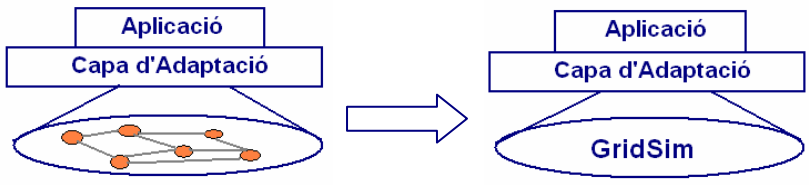


Fig. 4.9. Adaptació de l'aplicació a un GRID real.

Estudiant els tipus d'objectes que es necessiten crear, es determina que n'hi ha que són agrupables. Aquests objectes són el router, el GIS, l'usuari i el recurs. Llavors, ja que tots aquests elements en el fons són nodes de la xarxa GRID, i per tant, presenten una sèrie d'atributs i característiques similars, s'ajunten sota una interfície anomenada *NodeGrid*. La implementació d'aquesta interfície suposa l'obligatorietat d'implementar diversos mètodes comuns a tots els nodes (classes filles), com per exemple establir el seu nom, el tipus o la posició que ocupen sobre el mapa. A partir d'aquest plantejament, s'usen els conceptes de polimorfisme per a instanciar i crear els objectes. Així doncs, moltes funcionalitats (mètodes) en les que és indeferent conèixer de quin tipus específic d'objecte es tracta, es defineixen en un nivell superior en la jerarquia de classes, de manera que les classes finals sols han d'implementar els mètodes específics i concrets.

A més del polimorfisme, també s'aplica l'herència de classes. En aquests casos, s'ha usat l'herència directe com també l'herència per mitjà de classes abstractes depenent del cas. Per exemple, en el cas del GIS, GridSim defineix una classe abstracte, que ha servit per implementar els GIS amb les funcions específiques de cerca de cada algoritme.

4.5. Topologia de Xarxa

4.5.1. Objectius i Característiques

La topologia de xarxa és la capa formada per routers i els enllaços entre ells per tal de dotar de connectivitat extrem a extrem a l'escenari i permetre així la interconnexió de tots els elements que el formen, tant a nivell de xarxa com d'aplicació. A més, és la passarel·la que transporta els missatges entre els nodes d'aplicació. És important que la topologia de xarxa sigui consistent, és a dir, que compleixi els seus objectius de facilitar la transmissió de missatges. Per tant, els objectius que ha complir són:

- servir de xarxa de connexió entre elements de capa d'aplicació, anàlogament a qualsevol xarxa real, com és el cas d'Internet,
- permetre la connectivitat extrem a extrem,
- definir rutes entre els elements que siguin el més curtes possibles, és a dir, ha d'introduir algun tipus d'algoritme d'enrutat,
- definició de característiques pròpies d'enllaços de comunicacions, com retard de propagació, processat de missatges, o amplada de banda,
- impedir bucles infinits entre dos elements de la xarxa o més.

4.5.2. Configuració de la Topologia de Xarxa: Integració de BRITE

Un dels primers objectius és aconseguir generar topologies de xarxa que simulin el millor possible la configuració de xarxes reals que es poden trobar a

Internet. A més, aquest procés hauria de ser automàtic i amb capacitat de configurar diferents tipus de topologies a partir d'uns certs paràmetres d'entrada. Un paquet software que col·labora a donar una solució és BRITE, el qual s'ha integrat dins de l'aplicació desenvolupada. Tal com s'ha comentat a l'apartat 3.3. , BRITE entrega com a resultat de la seva execució un mapa de topologia de xarxa, format per routers i enllaços que els connecten, juntament amb les característiques d'amplada de banda i retard de cada un d'ells, així com les coordenades (x, y) que ocupen. A partir d'aquesta sortida de dades, s'ha de muntar l'entorn de simulació amb GridSim, juntament amb els objectes propis desenvolupats a l'aplicació.

És per tant important veure de quina forma es pot integrar el motor BRITE amb l'aplicació del projecte. La interfície gràfica (GUI) i la generació de la topologia, a nivell de codi, estan totalment separats, de forma que la interfície gràfica només s'encarrega de modificar el fitxer d'entrada al motor de BRITE, i aquest, per la seva part, sols interactua amb la GUI per actualizar l'estat de la generació que duu a terme.

Així doncs, la integració del motor de BRITE a l'aplicació desenvolupada sembla relativament clara, i sols és necessari trobar el punt de sortida on el generador entrega les dades de la topologia generada. Després d'investigar l'arbre UML de BRITE (veure **Fig. 8.23**), així com el codi font, es determina que el punt des d'on extreure les dades necessàries és qualsevol de les classes del package *Export*. Aquestes classes s'encarreguen de volcar a fitxer la topologia generada, donant-li el format corresponent i per tant, tenen accés a tot el que ha generat BRITE.

Llavors, el que s'ha fet ha estat modificar la classe de BRITE *Export.briteExport*, per aprofitar els mètodes d'extracció de dades, fent que l'esmentada classe modificada sigui el principal *input* de l'aplicació desenvolupada, i a partir del qual es genera la topologia IP amb objectes propis.

4.5.3. Generació de la Topologia de Xarxa

BRITE s'encarrega de configurar la topologia de xarxa, o xarxa a nivell IP, però falta transposar les dades que entrega en objectes de l'entorn de simulació, per tal de generar definitivament la topologia. Per tant, una vegada que tots els routers desitjats estan generats, juntament amb els enllaços de dades, es converteixen a objectes propis de l'aplicació PFC i s'integren dins de l'entorn de simulació. Aquest objecte és el *RouterGrid*, que és una classe *factory,* la qual retorna la instanciació dels diferents tipus de routers definits.

Inicialment, es va usar la classe *RouterRIP* definit per GridSim, la qual implementa una adaptació de l'algoritme d'encaminament RIP. Però immediatament es varen detectar mancances importants en la classe esmentada, com:

- falta d'escalabilitat a l'hora de crear la topologia mitjaçant RIP a mesura que la xarxa es va fent més gran,

- diversos errors en l'enviament dels paquets d'intercanvi de taules d'encaminament, els quals impedien la convergència de l'algoritme, ja que les taules d'encaminament s'enviaven sempre que un paquet RIP entrava a l'objecte i no sols quan hi havia un canvi en les mateixes.

Així doncs, a part de reimplementar el protocol RIP i no usar el plantejat per GridSim, es va pensar en variar el mode de generació de topologia usat fins ara, passant a definir una topologia IP seccionada en múltiples sistemes autònoms (AS), on dins de cada un d'ells es defineixen un determinat número de routers. D'aquesta forma, l'escalabilitat del simulador per a generar la topologia IP mitjançant RIP no es veuria tan efectada en funció del tamany de la xarxa IP.

S'ha de tenir en compte que RIP és un protocol d'encaminament vàlid per a xarxes relativament petites, on cada un dels routers conèix tota la topologia que l'envolta en un radi de fins a un màxim de 15 i que funciona mitjançant un intercanvi exhaustiu de les taules d'encaminament en mode de *flooding*. És a dir, el número d'entrades de la taula d'encaminament RIP és directament proporcional al número de routers de la xarxa. A més, per a una xarxa gran, amb un radi superior a 15 salts, RIP no és vàlid ja que incorpora un TTL amb aquest número de salts. És a dir, RIP no pot conèixer altres routers que estiguin més lluny de 15 salts d'ell mateix.

La separació de la topologia IP en diferents AS amb menor número de routers cada un, fa que les taules d'encaminament dels routers siguin menors: proporcionals al número de routers de dintre de la zona més un número limitat de routers frontera dels sistemes autònoms directament connectats. Així doncs, l'encaminament es simplifica i el temps necessari per a generar les taules de routing i per tant, per a que l'algoritme convergeixi, es veu sensiblement reduït. A més, desapareix el problema del radi de coneixement de RIP, ja que dins de cada una de les zones no serà possible que dos routers estiguin separats per més de 15 salts.

No obstant, aquesta separació en zones, introdueix una complicació extra a l'enrutat: s'han de definir rutes entre les zones. Aquest problema es resol mitjançant tres tècniques:

- la definició de routers frontera o passarel·la,
- l'interconnexió (enllaços) entre aquests routers i,
- la presa de decisions d'enrutat en els routers frontera.

En aquest entorn, un router frontera és aquell que interconnecta dos o més sistemes autònoms entre sí, i per tant, és la porta d'entrada/sortida de missatges procedents/destinats a d'altres sistemes autònoms. En certa manera, es poden imaginar com a routers d'agregació, com si es tractàs de

routers de diferents operadors que s'interconnecten per tal d'intercanviar directament el tràfic dels seus clients.

La particularitat que tenen els routers frontera és la de no publicar rutes internes de les zones a les que pertanyen, de manera que cada zona es troba aïllada de les demés (**Fig. 4.10**). Així doncs, només cada zona de forma interna coneix la seva propia topologia, mentre que no coneix absolutament res de les zones externes, excepte els routers frontera directament connectats.

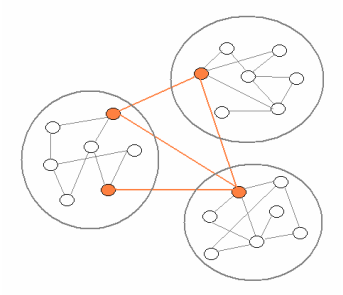


Fig. 4.10. Connectivitat entre sistemes autònoms.

Els routers frontera s'han de connectar entre ells mitjançant enllaços físics. Aquesta connexió la proporciona el mateix BRITE, usant el mode de connexió entre AS. Cal fer notar que cada router frontera només coneix els routers frontera que hi estan directament connectats, ja que no s'ha implementat cap algoritme de routing a la zona "interAS", perquè complicava sense motius la generació de la topologia, i el benefici extret en la reducció del número de salts entre AS no es veia recompensada per l'esforç d'implementació.

Com que es probable que no tots els sistemes autònoms coneguin directament la resta d'AS que formen el conjunt de la xarxa, s'han de prendre decisions d'enrutat, enviant missatges amb destí a un sistema autònom desconegut, usant-ne un de conegut com a intermediari. En la **Fig. 4.11** es pot observar que un node de l'AS1, que vol enviar a un altre node que pertany a l'AS3, no hi té connexió directe, i que es veu obligat a usar el node frontera de l'AS2 com a intermediari dels missatges.

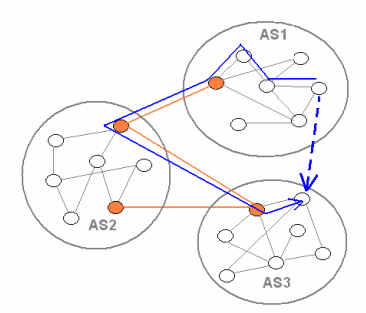


Fig. 4.11. Exemple de pas de missatges entre AS intermediari.

Així doncs, es poden definir clarament una sèrie de diagrames de fluxe per als dos routers que formen la topologia IP de la xarxa, explicatius de les tasques

que duen a terme i de les regles d'enrutat que implementen en cada cas. Els diagrames es poden trobar en l'annex 8.8.2.

4.6. Topologia d'Aplicació

La topologia d'aplicació es genera sobre mateix de la topologia de xarxa, fent que cada un dels nodes GIS del GRID estigui unit a un router de xarxa mitjançant un link amb unes certes característiques de retard i amplada de banda.

Es defineixen dos mètodes de distribució de GIS sobre la xarxa IP: distribució seqüencial i distribució aleatòria.

La distribució seqüencial (**Fig. 4.12** (2)) assigna, per ordre, cada un dels GIS a un router, fins que no n'hi han més per a repartir. És a dir, uneix per exemple el GIS0 amb el router0 i el GIS10 amb el router10. D'aquesta forma, la distribució és molt ràpida i senzilla però té l'inconvenient que si la xarxa IP és molt més gran (gran quantitat de routers) que el número de GIS a distribuir, els GIS queden concentrats en una determinada zona de la topologia, mentre que una gran part de la mateixa esdevé deserta de nodes d'aplicació.

Per tal de millorar la distribució dels nodes i que cobreixin gran part de la xarxa IP, es planteja una distribució aleatòria dels mateixos (**Fig. 4.12** (1)). Aquesta distribució es duu a terme asignant cada un dels GIS a un router escollit aleatòriament. Usant aquest mètode, els nodes es distribueixen uniformament a través de la xarxa, sense que quedin grans zones sense cap GIS.

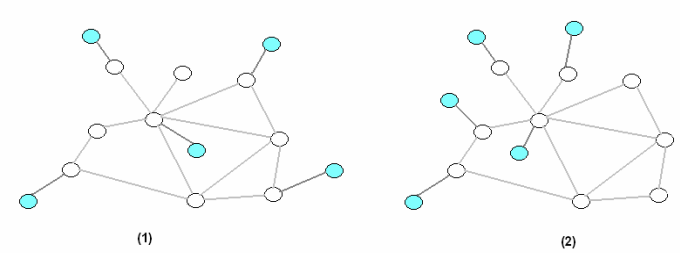


Fig. 4.12. Distribució de nodes. **(1)** Aleatòria. **(2)** Seqüencial.

Tal com s'ha explicat anteriorment, la topologia d'aplicació està formada per diferents entitats, cada una de les quals realitza les seves propies funcions. Seguidament es detallen les funcionalitats que implementen.

4.6.1. Implementació de GIS

Com la resta d'objectes desenvolupats, l'objecte GIS (*GisGrid*) implementa la interfície *NodeGrid*. Aquest objecte inclou les funcionalitats principals d'un GIS, com emmagatzemar el nom, l'identificador intern, l'AS al que pertany i mètodes per a la generació i evaluació dels GIS veïns depenent de l'identificador propi. A la vegada, cada un dels algoritmes de cerca està implementat estenent aquesta classe juntament amb la implementació de les funcionalitats de GIS del

toolkit GridSim (*AbstractGis*). És a dir, la classe *GisGrid* implementa les funcionalitats comunes de tots els GIS, mentre que d'aquesta hereden els GIS finals que implementen les funcions específiques de cada un dels algoritmes de cerca (veure 4.10.). La **Fig. 4.13** mostra la jerarquia de classes comentada. Per instanciar les classes s'usa una factory (*GisGridFactory*).

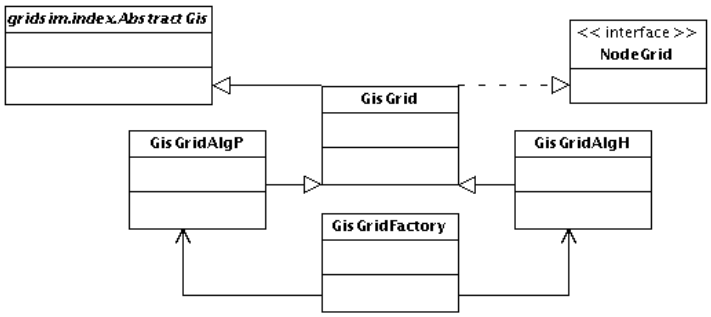


Fig. 4.13. Jerarquia de classes referents al GIS.

4.6.2. Implementació d'Usuaris

Anomenam usuari a la classe que s'encarrega de realitzar les funcionalitats que un usuari real realitzaria en una xarxa, és a dir, principalment el llançament de les cerques de recursos a través del GRID. L'objecte està connectat directament amb un router de la topologia de xarxa i passa a pertanyer a l'AS corresponent al router que li fa de porta d'accés al GRID. A més, se li assigna un GIS com a porta d'entrada al nivell d'aplicació del GRID. Aquest GIS s'anomena *GIS Regional*, i és el que reb en primera instància les peticions que l'usuari genera contra el GRID.

Tots els usuaris que es defineixen hereden de la classe *UsuariGrid*, la qual obliga a implementar les característiques imprescindibles dels usuaris. A la vegada, *UsuariGrid* implementa la interfície *NodeGrid* i estèn la classe *GridUser* de GridSim. Així doncs, d'aquesta forma un usuari es converteix en un *NodeGrid*, com la resta de nodes de l'aplicació i a més, hereda les funcionalitats de l'usuari *GridUser* de GridSim. El diagrama UML de la **Fig. 4.14** mostra la jerarquia de classes esmentada.

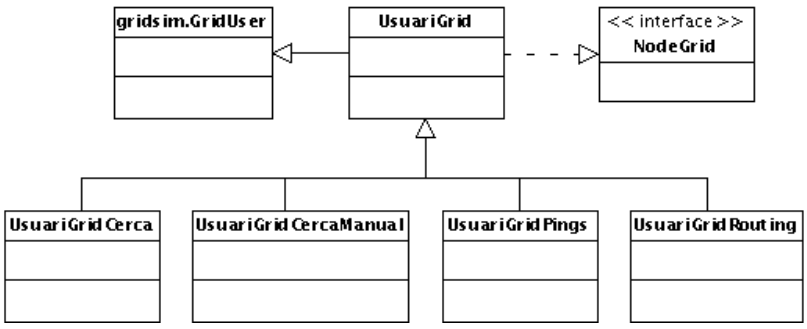


Fig. 4.14. Jerarquia de classes referents als usuaris.

Inicialment, un usuari és l'encarregat d'interactuar amb els elements de la xarxa, però se li poden agregar diverses funcionalitats. A més, l'usuari es defineix com un dels punts d'entrada claus per tal de definir el comportament de les simulacions.

Així doncs, a part d'encarregar-se d'iniciar el procés de cerca de recurs, també realitza els ajustaments probabilístics de recursos i d'activitat de GIS definits a 4.6.5. És a dir, des d'aquesta classe, es realitzen múltiples iteracions per a cada parella de probabilitat d'activitat de GIS i de probabilitat de recurs per a cada GIS, on cada una d'elles representa el llançament d'una cerca.

D'aquesta forma, sols s'ha de modificar aquesta classe per tal de modificar el conjunt de valors de probabilitats i és independent dels objectes generats anteriorment a la topologia IP i d'aplicació.

Es defineixen tres tipus diferents d'usuaris usables depenent del que es vulgui obtenir amb la simulació. Dins d'aquests tres tipus d'usuaris, es poden dividir dos grups: un primer grup d'usuaris que s'encarreguen de llançar processos de cerca i un altre grup útils per a realitzar funcions de comprovació i debug.

Usuari UsuariGridCercaManual

L'usuari *UsuariCercaGridManual*, tal com indica el seu nom, és un actor que llança processos de cerca de recursos. La particularitat que presenta és que defineix tots els GIS com a actius, excepte alguns de concrets, i el que es varia és sols la probabilitat que un GIS conegui un recurs. És a dir, fixa com a determinista l'estat dels GIS i com a probabilístic l'existència dels recursos. Aquest usuari és útil per a realitzar proves de cerca específiques controlant exactament en cada moment quins GIS estan actius i quins no.

Usuari UsuariGridCerca

L'*UsuariGridCerca* és la classe que implementa l'usuari més complet, on utilitza completament valors probabilístics tant d'activitat de GIS com de recurs contingut en cada un d'ells. Per a cada parella de valors de probabilitats d'activitat de GIS i de recurs, es realitzen 20 iteracions, en cada una de les quals es varia el marcatge d'activitat i de recurs, de la forma que s'explica a 4.6.5. El diagrama de fluxe de la classe es pot consultar en l'annex 8.8.1.

Altres Usuaris

Altres classes d'usuaris definits realitzen funcions de comprovació i de debug. En aquest conjunt d'usuaris, existeixen:

- l'usuari amb la capacitat de realitzar pings entre ell mateix i qualsevol node d'aplicació al llarg de la xarxa, per tal de comprovar la connectivitat dels elements del GRID i camí recorregut a mode de *traceroute*,

- l'usuari que és capaç d'extreure les taules d'encaminament de qualsevol router de la topologia IP implementada. L'usuari és útil per a realitzar tasques de comprovació de la topologia de la xarxa i veure el contingut de les taules de routing per tal de detectar possibles errades en l'execució dels algoritmes d'encaminament.

4.6.3. Implementació de Recursos

En l'entorn GRID, un recurs s'anomena a aquell element que forma part de la xarxa i el qual posa a disposició dels usuaris els seus recursos de processament o d'emmagatzematge d'informació a través de la infraestructura de GRID.

GridSim defineix el que es un recurs a la classe *GridResource*. En ella s'hi poden incloure característiques dels recursos de computació disponibles, com per exemple, potència de la CPU, tamany de memòria RAM, espai de disc disponible, etc, inclús definir polítiques de planificació i de reserva dels mateixos. No obstant, totes aquestes característiques no són importants per a l'anàlisi dels algoritmes de cerca implementats i per tant, són dades supèrflues i que no aporten res als resultats i a les corresponents conclusions.

Això és perquè en el nivell del present projecte, és completament anàleg cercar un recurs amb "x CPU, y memòria i z disc lliure", que cercar el recurs anomenat específicament "1". A més, l'únic que canvia entre un cas i l'altre és la lògica implentada en els GIS. En el primer cas, s'han de realitzar cerques condicionades dins de la llista de recursos que coneix, mentre que en el segon cas, comprovar l'existència o no del recurs sol·licitat és tan fàcil com consultar directament una llista d'identificadors.

Així doncs, un recurs serà simplement un identificador que s'emmagatzema en un llistat que pertany a cada GIS. Una vegada que en un GIS es reb una petició de cerca de recurs, el que ha de fer és simplement executar una comanda tipus *array.contains(identificador)*, tornant *true* o *false* segons la seva existència.

4.6.4. Paquet de Dades d'Aplicació

Per a que les dades de nivell d'aplicació puguin ser intercanviades entre els diferents actors que formen part de l'escenari, és imprescindible crear algun tipus de mitjà per empaquetar-les i que siguin fàcilment consultades. Per això, s'ha creat un paquet de nivell d'aplicació, que transporta la informació dels processos de cerca que llencen els usuaris.

El paquet està implementat per la classe *GISPacket*, que com el seu nom indica, és el que s'intercanvien els GIS entre sí i és també el missatge que l'usuari sol·licitant d'un cerca d'un recurs usa per posar-se en contacte amb el GIS Regional que li correspon.

El *GISPacket* conté dades referents als processos de cerca, així com informacions estadístiques del recorregut que segueix a través del GRID. Els elements que guarda *GISPacket* són:

- **Camí d'aplicació**, on s'hi emmagatzemen els salts que realitza el paquet des de que és generat per l'usuari atravessant la topologia del GRID.

- **Camí de xarxa**, en el qual s'hi guarden tots els salts que realitza el paquet fins arribar al seu destí, comptabilitatzant tant els salts de xarxa com els d'aplicació.

- **Temps consumit**, referenciat al cami seguit, i indica els instants de pas del paquet a través dels routers, GIS i usuaris. Per tant, quan el paquet ha recorregut tot el camí, la diferència entre l'instant inicial del llançament del paquet i l'instant de la seva destrucció denota el temps de viatge del mateix a través de la xarxa.

- **GIS on s'ha trobat el recurs sol·licitat**, indicant l'identificador del GIS que conté el recurs sol·licitat.

- **Identificador de paquet**, identificador únic i incremental que es genera a l'hora de construir inicialment el paquet.

- **Camp de TTL**, que conté el número de salts que resten per a que el paquet sigui descartat per la xarxa, per haver excedit el límit de salts, i així evitar possibles events de bucles infinits.

- **Recurs buscat**, variable que conté l'identificador del recurs que l'usuari està buscant.

4.6.5. Probabilitats d'Activitat i de Recurs Present

Per tal d'automatitzar les condicions de l'escenari de simulació i realitzar estudis estadístics del comportament dels algoritmes de cerca, es defineixen dues classes per a marcar l'estat d'activitat dels GIS, així com determinar, per a cada GIS, si conèix o no un determinat recurs.

L'estat del GIS es refereix a que si un determinat GIS es troba o no disponible per acceptar i processar peticions de cerca. L'estat pot ser transitori, és a dir, que puntualment un GIS canvii el seu estat d'actiu a no actiu pel fet d'estar sobrecarregat, o també pot ser definitiu, fent que no estigui disponible contínuament perquè s'ha desconnectat o apagat. En l'entorn de simulació, els GIS es mantenen en el mateix estat d'activitat durant cada iteració, i es remarquen tots ells abans d'executar-ne una, depenent de la probabilitat d'activitat de GIS de la iteració.

Per la seva part, cada un dels GIS pot conèixer o no un o més recursos, és a dir, que davant d'un procés de cerca pot respondre positiva o negativament a la petició del recurs per part de l'usuari consultant el registre de recursos

coneguts. De la mateixa manera que en el marcatge dels GIS actius, els recursos es redistribueixen per a cada iteració depenent de la probabilitat d'existència del mateix.

En el cas del marcatge dels GIS, els mètodes que es defeneixen són els següents:

- **Marcatge manual**. Es decideixen manualment quins són els GIS que esdevenen inactius. Per a fer-ho, es pasen els identificadors dels GIS que es volen marcar com a inactius.

- **Marcatge aleatori**. Marca els GIS no actius depenent de si la probabilitat d'entrada (entre 0 i 1) demanada és major o menor que un número aleatori generat (també comprès entre 0 i 1). Llavors, només si la probabilitat demanada és menor que el número generat, el GIS esdevé com a inactiu. D'aquesta forma, com major sigui la probabilitat demanada, major és el número de GIS actius.

- **Marcatge aleatori acotat a un número**. En aquest mètode, es recorren tots els GIS generats, de forma aleatoria, i es marquen successivament com a inactius fins que en total s'ha marcat el número desitjat de GIS.

- **Marcatge per funció probabilística**. Aquest mètode està pensat per a marcar els GIS a partir d'una funció de probabilitat, com per exemple una funció gaussiana, normal, exponencial, etc.

Per la seva part, els mètodes de marcatge dels recursos disponibles en cada GIS són anàlegs als que s'han comentat a sobre, adaptat-los al recursos. En aquest cas, el que es fa és incloure un recurs com un identificador dins d'una llista de recursos propi de cada un dels GIS.

Per defecte, els GIS es generen inicialment com actius i sense cap recurs conegut. Per tant, si es vol que hi hagin GIS inactius i recursos distribuïts a través del GRID, s'han d'aplicar aquestes dues classes. La modificació tant de l'activitat dels GIS com dels recursos que conèixen es pot fer en temps de simulació, sense haver de regenerar la topologia, i precissament es realitza dins de la lògica dels objectes usuari desenvolupats.

4.7. Dades de Topologia

Totes les dades necessàries per a l'execució de cada una de les iteracions que composen les simulacions s'emmagatzemen en una única classe, que adopta el format de *singleton*, és a dir, només existeix un únic objecte d'aquesta classe, i és impossible instanciar-ne més d'un. D'aquesta forma, totes les dades es troben centralitzades en un únic punt i es pot assegurar que no es dupliquen ni es corrompen, i a més, són fàcilment accessibles i modificables mitjançant mètodes *get* i *set*.

Per una part, emmagatzema un mapa complet de la topologia, amb tots els nodes que la conformen (GIS, routers i usuaris), indexat per una clau única

consistent amb l'identificador de cada objecte. Mitjançant aquest mapa, l'accés a tots els objectes que formen la topologia és molt ràpid.

A més del mapa complet anterior, es defineixen una sèrie de llistes per a cada un dels elements de la topologia, on només s'hi guarden els objectes de cada tipus, el que facilita la separació dels diferents d'elements i el seu posterior ús.

A part de les dades emmagatzemades, es defineixen mètodes de consulta complexes, com retornar una llista dels GIS que es troben dins d'un determinat AS o determinar el número de recursos distribuïts al llarg de tota la topologia.

4.8. Estadístiques de Simulació

De la mateixa forma que les dades de topologia, les dades resultants de l'execució de les simulacions s'han de guardar per a posteriorment realitzar-li tractaments estadístics. Així doncs, les estadístiques es guarden per a cada una de les iteracions o procés de cerca individual i es volquen a un fitxer de text, el format del qual es pot consultar a 8.12. És a dir, després de cada iteració, s'eliminen de memòria totes les dades generades, de manera que s'evita l'ús innecessari de RAM per emmagatzemar dades que no tornaran a ser usades internament dins de l'aplicació.

De la mateixa forma que el contenidor de dades de la topologia, les estadístiques es guarden en un classe (*EstadistiquesGrid*) que compleix el patró de disseny de *singleton*. Internament, aquesta classe és un *HashMap* indexat per un identificador únic que identifica cada procés de cerca. La parella de dades d'aquesta clau són les dades estadístiques del corresponent procés de cerca, més concretament, el llistat dels paquets d'aplicació (*GISPacket*) que s'han generat dins pel procés de cerca. Específicament, aquestes dades estan organitzades en una classe interna anomenada *EstadistiquesGrid*. Amb aquesta llista, es té tota la informació necessària per a generar estadístiques, les més importants de les quals són:

- **Número de missatges d'aplicació** generats per a cada procés de cerca.

- **Número de branques de cerca obertes** dins del GRID, entenent com a branques els camins independents amb els que es divideix un procés de cerca.

- **Número de salts d'aplicació** de cada una de les branques de cerca obertes.

- **Número de salts de xarxa** realitzats per a cada procés de cerca.

- **Els instants de temps d'entrada i sortida** dels paquets del procés de cerca quan atravessa cada un dels elements de xarxa (routers o GIS). El temps total d'un procés de cerca es determina realitzant la diferència enter el l'instant d'entrada al darrer GIS consultat i l'instant inicial en el qual l'usuari llença el procés de cerca.

- **Llistat dels recursos trobats** durant la cerca al llarg del GRID.

4.8.1. Volcat d'Estadístiques a Fitxer

Com s'ha esmentat, les dades estadístiques es volquen dins d'un fitxer mitjançant l'ús de la classe *ExportEstadistiques*. El format del fitxer es troba a 8.12. , i el procés per a volcar les dades és el de consultar el *singleton EstadistiquesGrid* i escriure a fitxer de forma ordenada les mètriques interessants. Al final del fitxer, s'escriuen les funcions necessàries per a que una fulla de càcul calculi els valors màxims, mínims, mitjanes i desviacions típiques de cada una de les mètriques esmentades anteriorment. D'aquesta forma, només s'ha de carregar l'arxiu de text a la fulla de càlcul i automàticament s'obtenen els valors estadístics per a cada simulació.

4.9. Visualització

Una opció disponible com a entrada per paràmetre a l'hora d'executar l'aplicació desenvolupada, és el de triar si es generen arxius de mapa de topologia amb el format d'entrada a l'aplicació Otter. Aquests arxius contenen les dades de topologia IP i d'aplicació de cada iteració simulada, i una vegada que es carreguen a Otter, es visualitzen en format gràfic. A més, és possible mostrar els camins d'aplicació que recorre cada un dels missatges, i així comprovar gràficament el recorregut que realitzen les branques de cerca dins del GRID. El format del fitxer d'entrada a Otter es descriu a l'annex 8.11.

4.10. Implementació dels Algoritmes de Cerca

La implementació dels algoritmes de cerca s'ha realitzat seguint les especificacions corresponents exposades a 2.4.3. per l'algoritme P i a 2.4.4. pel cas de l'algoritme H. Tal com s'esmenta a 4.6.1. quan es parla de la generació dels GIS, els algoritmes estan implementats com un GIS que incorpora la lògica de cada un dels algoritmes.

Per l'algoritme P, es crea la classe *GisGridAlgP*, la qual extén les funcionalitats de la classe pare *GisGrid*. Respecte de l'algoritme original, es realitza la modificació de incorporar-hi la funcionalitat d'evaluar si el node està actiu o no. Si està inactiu, el missatge es processa com a rebut però no realitza cap altra acció.

Pel que fa a l'algoritme H, comentar que es crea la classe *GisGridAlgH*, filla de *GisGrid* i que implementa la lògica de l'algoritme. Com en el cas de l'algoritme P, també es realitza el reconeixement d'activitat del propi GIS i si és inactiu, el missatge es reb i es guarda però no s'hi realitza cap altra acció. No obstant, s'ha hagut de realitzar una modificació a l'algoritme inicial. La línia 16 de la **Fig. 8.22** de l'annex 8.4.2. es substitueix pel següent pseudocodi:

IF ((v_d[k] = dalive) AND (non-alive > 1)) THEN

Això és així ja que encara que $v_d[k]$ = *dalive* es compleixi, si el nombre de veïns no actius no és major que 1, no es disposa de la variable v_{a2}, i per tant, no ha d'entrar dins d'aquesta condició.

Finalment, per a determinar l'estat d'activitat dels nodes GIS veïns, es consulta directament una variable booleana a través de l'API de GridSim, i es reorganitza el vector v_d tal com indica l'algoritme. És a dir, la consulta de l'estat dels veïns no incrementa el retard en l'algoritme.

4.11. Verificació de la Implementació

4.11.1. Plantejament

Per a comprovar la implementació i el correcte funcionament de l'aplicació desenvolupada es planifiquen una sèrie de verificacions. La primera consisteix en comprovar que existeix connectivitat entre l'usuari i qualsevol node d'aplicació del GRID. Per a fer-ho, s'executa la funció de ping, observant com el paquet generat per l'usuari recorre tota la xarxa fins al seu destí i retorna al seu origen sense cap problema. Si l'eïna ping falla en algun punt, es determinaria que la implementació no permet l'intercanvi de missatges entre dos nodes. En cas contrari, es podrà afirmar que la implementació ha estat correcte. La comprovació es troba en l'annex 8.13.

Per altra banda, és crític realitzar una verificació del funcionament dels dos algoritmes de cerca implementats. En ambdós casos, se suposa que els nodes veïns del GIS0 (node inicial del procés de cerca) en les dimensions 0 (GIS1), 2 (GIS4) i 3 (GIS8) estan inactius i que, a més, cap dels nodes del GRID posseix el recurs que és objecte de la cerca.

Aquesta verificació es realitzarà comparant el resultat obtingut a partir de l'anàlisi teòric front els resultats pràctics oferts per l'aplicació, considerant en els dos casos la configuració esmentada de GIS actius i inactius, tot usant *UsuariGridCercaManual* (4.6.2.). Si la implementació és correcte, llavors els dos resultats han de ser idèntics. El resultat pràctic es mostra en forma de captures de la topologia dibuixada per Otter, tot marcant el recorregut dels diferents missatges d'aplicació al llarg del GRID. El pas dels missatges es marca amb un traç negre gruixut, mentre que cada branca quan es selecciona individualment es remarca amb el mateix traç però de color blau. Finalment, al nom dels nodes actius se'ls hi afegeix el sufix "a", mentre que als inactius no se n'hi afegeix cap.

4.11.2. Verificació del Funcionament de l'Algoritme P

Teòric

Seguint l'algoritme de l'apartat 2.4.3. , implementat en 4.10. , el resultat teòric de la cerca és el que apareix en **Fig. 4.15**.

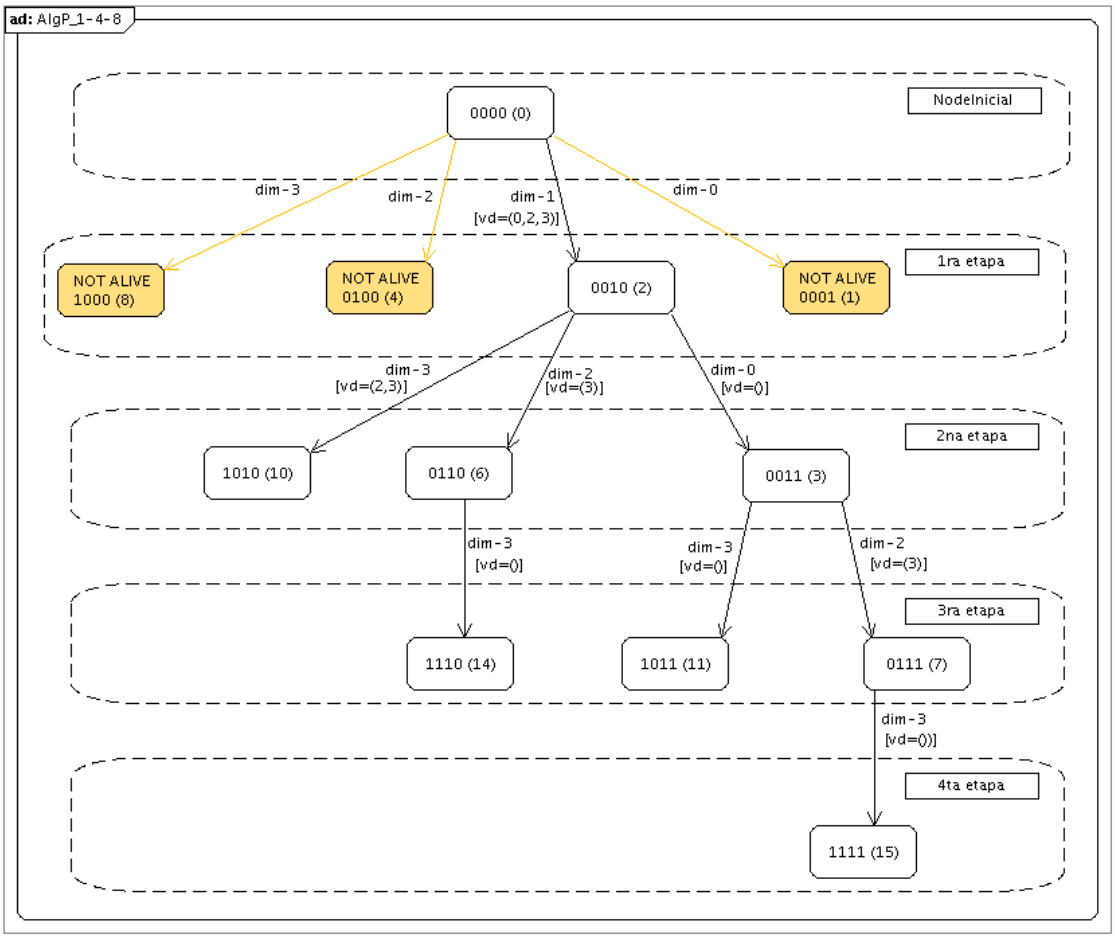


Fig. 4.15. Exemple teòric de l'execució de l'algoritme P.

Els nodes de color groguenc es corresponen amb els nodes inactius.

La **Taula 4.2** mostra, de forma ordenada, la mateixa informació que la **Fig. 4.15** anterior. En ella s'indiquen les dades referents a cada branca de cerca oberta dins del GRID, expressant el node final on acaba la mateixa, el camí que recorre i el número de salts d'aplicació que ha realitzat.

Branca ID	*Node final*	*Camí aplicació*	*Salts aplicació*
1	G10	G0 – G2 – G10	2
2	G11	G0 – G2 – G3 – G11	3
3	G14	G0 – G2 – G6 – G14	3
4	G15	G0 – G2 – G3 – G7 – G15	4

Taula 4.2. Recorregut branques cerca per algoritme P.

Pràctic

Pel que fa referència al resultat pràctic, la **Fig. 4.16** mostra la composició del camí establert per la BrancaID 4.

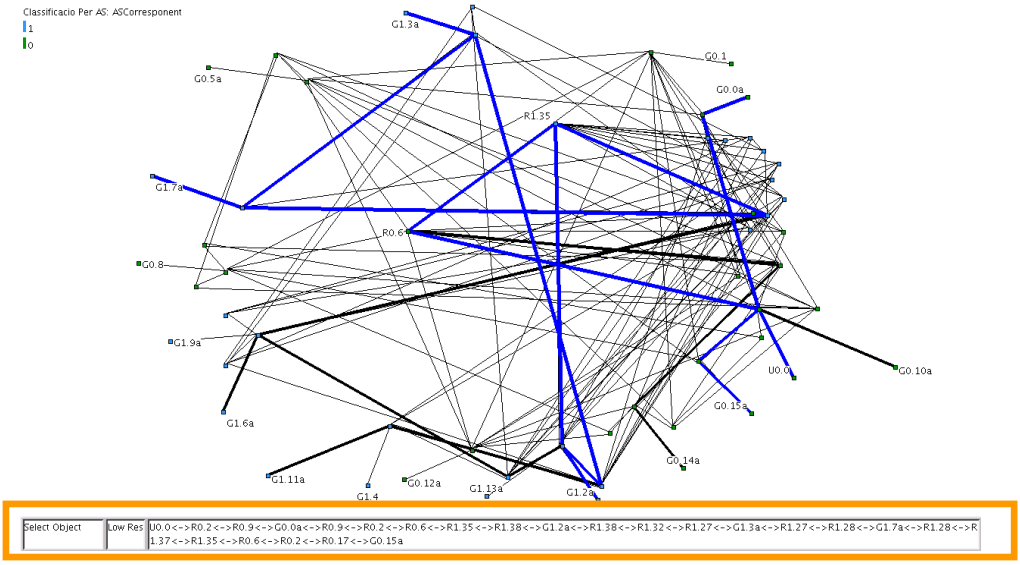


Fig. 4.16. Recorregut de la BrancaID 4 mostrat per Otter.

En la **Fig. 4.16** es mostra remarcat el recorregut de la branca sel·leccionada. Es comprova de forma immediata com efectivament el recorregut de la branca 4 és idèntic tant pel cas teòric com pel pràctic. Les figures corresponents a la resta de missatges generats pel procés de cerca es troben a l'annex 8.14.1.

En total, tant en el cas teòric com en el pràctic, l'algoritme ha realitzat, per la branca de cerca més llarga, 4 salts d'aplicació, valor que es correspon amb les dimensions de l'hipercub, tal com enuncia l'algoritme P, fet que ve a demostrar el correcte funcionament de la implementació.

4.11.3. Verificació del Funcionament de l'Algoritme H

Teòric

Seguint l'algoritme de l'apartat 2.4.4. i la corresponent implementació 4.10. , el resultat teòric de la cerca per al cas plantejat és el plasmat en la **Fig. 4.17**.

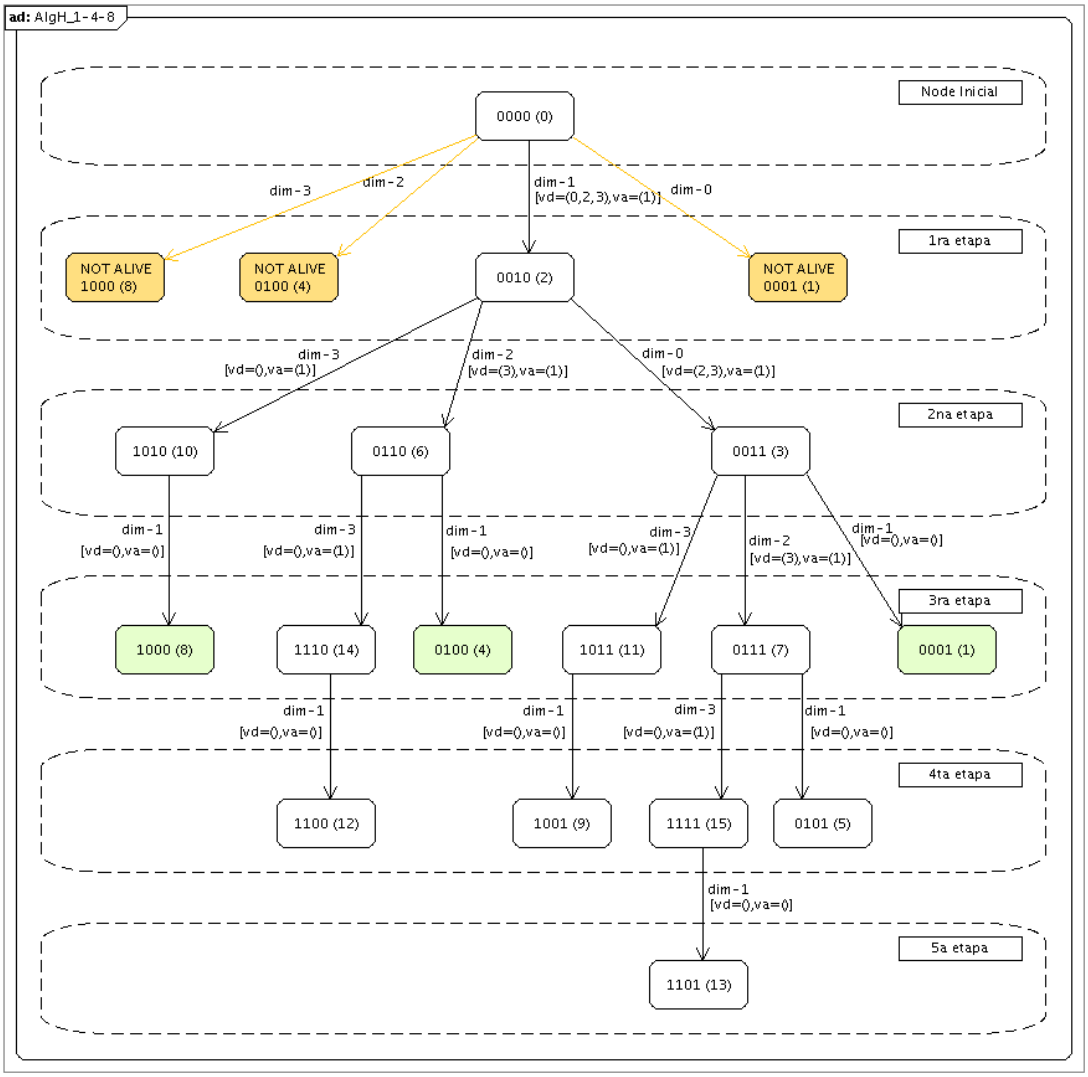


Fig. 4.17. Exemple teòric de l'execució de l'algoritme H.

Els nodes marcats amb color groguent són els GIS inactius, mentre que els marcats amb verd són els mateixos inactius alcançats des d'altres camins, deixant que siguin simples nodes finals de branques.

La **Taula 4.3** mostra de forma resumida, a partir de la **Fig. 4.17**, el camí a nivell d'aplicació de les branques de cerca obertes dins del GRID:

Branca ID	*Node final*	*Camí aplicació*	*Salts aplicació*
1	G1	G0 – G2 – G3 – G1	3
2	G5	G0 – G2 – G3 – G7 – G5	4
3	G13	G0 – G2 – G3 – G7 – G15 – G13	5
4	G9	G0 – G2 – G3 – G11 – G9	4
5	G4	G0 – G2 – G6 – G4	3
6	G12	G0 – G2 – G6 – G14 – G12	4
7	G8	G0 – G2 – G10 – G8	3

Taula 4.3. Recorregut branques de cerca per algoritme H.

Pràctic

Si s'analitza el resultat pràctic, en la **Fig. 4.18** es dibuixa el missatge que recorre la *BrancaID=3* (veure **Taula 4.3**):

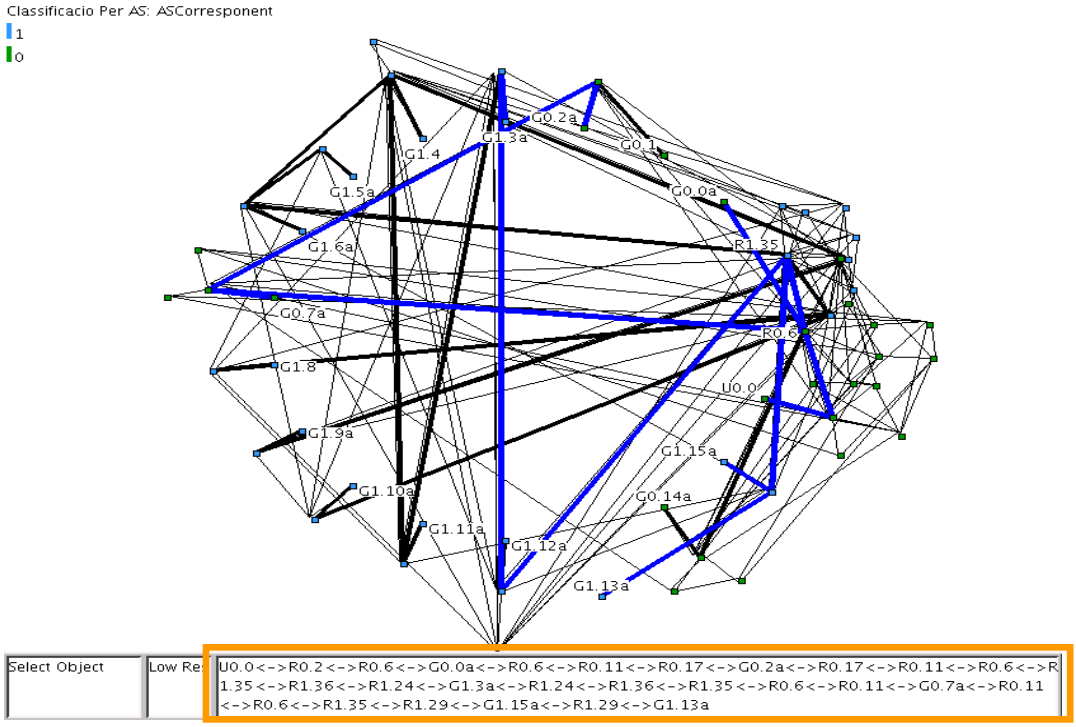


Fig. 4.18. Recorregut de la BrancaID 3 mostrat per Otter.

Es pot comprovar com efectivament el camí que segueix el missatge d'aplicació al llarg de la *BrancaID=3* és el mateix que l'expressat a la **Taula 4.3**. La llista completa de branques amb els corresponents salts d'aplicació es troben a l'annex 8.14.2.

Si s'observa amb deteniment la **Fig. 4.18**, a diferència del que passa amb l'algoritme P, es pot comprovar com tots els nodes GIS actius són consultats durant el procés de cerca, sense que els tres nodes inactius ho impedeixin. A més, aquests nodes inactius tornen ser consultats des d'altres camins i esdevenen com a simples nodes finals de branca.

Tenint en compte que l'algoritme H assegura que per arribar a tots els nodes fan falta com a màxim $r\dim+1$ salts, es pot assegurar que aquest exemple pràctic ha produït els mateixos resultats que la realització teòrica del mateix i que s'ajusten al que defineix l'algoritme H. Per tant, es determina que la implementació de l'algoritme és satisfactòria.

CAPÍTOL 5. SIMULACIONS

5.1. Objectius

Els objectius que es busquen obtenir a partir de les simulacions són els següents:

- Comprovar la independència dels algoritmes de cerca implementats front a la xarxa IP sobre la qual es genera.
- Comprovar la relació entre el número de missatges intercanviats front la dimensió de l'hipercub.
- Comprovar la relació entre el número de salts a nivell IP front als salts a nivell d'aplicació depenent de la dimensió de l'hipercub.
- Determinar l'existència de relació entre la probabilitat de GIS actiu i el nombre de recursos trobats al llarg del GRID depenent de la dimensió de l'hipercub.
- Determinar la possible relació entre el número de missatges i el número de branques de cerca obertes depenent de la dimensió de l'hipercub.

5.2. Metodologia

Les simulacions es realitzen sobre dos equips amb les següents característiques tècniques (Taula 5.4):

Característica	*Equip 1*	*Equip 2*
CPU	P4 2.6GHz	C2D 1.6GHz
Memòria RAM	1.5Gb	2Gb
Disc Dur	1x120GB + 1x160GB	1x160GB

Taula 5.4. Característiques dels equips de simulacions.

Per tal de llançar cada un dels grups de simulació, es ideal escriure un script bash que executi automàticament les comandes necessàries per tal d'iniciar cada una de les simulacions parcials, i a més, configurar l'arxiu d'entrada del generador de topologia IP. Aquest punt s'amplia en 8.10.

Per la seva part, els paràmetres d'entrada per línia de comandes de l'script es troben a 8.17.

Per a cada una de les dimensions considerades, es realitzen successius processos de cerca variant progressivament les probabilitats d'activitat de GIS i les de recurs conegut per GIS, tal com s'explica a 4.6.5. , però imposant l'execució d'un únic algoritme de cerca. Per a cada parella de probabilitats, es

llencen 20 iteracions idèntiques per tal de generar estadístiques mitjanes a partir d'un grup de mostres suficient. Així doncs, en total, per a cada procés llençat, es realitzen un total de 2200 iteracions. Per tant, per a cada dimensió es generen dos processos de cerca (un per a cada algoritme de cerca), el que resulta ser 4400 iteracions.

En total, si es sumen el total d'iteracions per a cada un dels processos, es determina que la simulació té un cost 44000 iteracions o processos de cerca, que aproximadament suposen unes 6-7 hores de simulació sobre l'equip 1.

5.3. Primer Grup de Simulacions

5.3.1. Configuració

En aquest primer grup de simulacions, l'aplicació genera una xarxa IP amb un total de 100 routers, distribuïts en 10AS, això és, 10 routers per a cada AS. Sobre aquesta xarxa IP, es generen successives topologies hipercub que van des de dimensió 1 (2 nodes GIS) fins a dimensió 10 (1024 nodes GIS). Seguidament es duplica el nombre de routers però es manté el número d'AS i es tornen a realitzar les simulacions.

Resumint, la **Taula 5.5** mostra els paràmetres del grup de simulacions.

Núm. Total routers	*Intercon. entre routers*	*Núm AS*	*Intercon. entre AS*	*Dimensió hipercub*	*Probabilitat GIS actiu*	*Probabilitat recurs conegut*
100	4	10	6	1,2,..., 10	0.1,0.2,...,1	0,0.1,0.2,...,1
200	4	10	6	1,2,..., 10	0.1,0.2,...,1	0,0.1,0.2,...,1

Taula 5.5. Configuració de les simulacions de Grup1.

En l'annex 8.15. es poden consultar les dades extretes a partir de l'execució de les simulacions. A continuació es realitza l'anàlisi de les mateixes.

5.3.2. Anàlisi de les Dades

En l'annex 8.16. es troben els anàlisis extesos de les dades contingudes a les taules 8.15. En aquest apartat sols es plasmaran les conclusions més importants extretes dels esmentats annexes. Així doncs, el següent llistat enumera les conclusions clau:

- **Número de missatges d'aplicació acotat a 2^n**, on *n* és la dimensió de l'hipercub. Per a una cerca, no es generen mai més de 2^n missatges, valor que es correspon amb el número de nodes GIS distribuïts. El màxim de missatges succeeix quan la probabilitat d'activitat dels GIS és 1 i la probabilitat de recurs conegut és 0. És a dir, tots els nodes són actius i no hi existeix cap recurs dins el GRID, de manera que els algoritmes recorren íntegrament la topologia d'aplicació.

- **Número de branques de cerca limitat a 2^{n-1}**, on *n* és la dimensió de l'hipercub. Per a una cerca, com a màxim s'obren la meitat de branques que de nodes GIS. Aquest cas passa si i només si tots els GIS estan actius i no hi ha cap recurs dins el GRID.

- **Número màxim de salts d'aplicació del camí més llarg està acotat**. Aquest llindar es correspon exactament amb la descripció de l'algoritme corresponent (veure 2.4.3. i 2.4.4.) és a dir:

  - En cas de l'algoritme P, la longitud màxima del camí és *n*.

  - Per l'algoritme H, la longitud màxima del camí és $n+1$.

- **S'assegura que el recurs es troba si** $n \geq 5$. Per a hipercubs amb dimensió estrictament menor a 5, no es pot assegurar que en mitjana, independentment de l'algoritme de cerca, es trobi el recurs sol·licitat.

- Fixada la dimensió de l'hipercub, amb probabilitat d'activitat de GIS baixa, l'algoritme H tramet major nombre de missatges d'aplicació (**Fig. 5.19**) i obri més branques de cerca, el que a la vegada implica l'acumulació de major nombre de salts IP front de l'algoritme P amb les mateixes condicions. A mesura que la probabilitat d'activitat de GIS augmenta, les mètriques s'assemblen més. Amb aquestes condicions, l'algoritme H troba més recursos, és a dir, amb molts GIS inactius (probabilitat d'activitat baixa) té un millor comportament que l'algoritme P. La resta de gràfiques es poden consultar a l'annex 8.16.

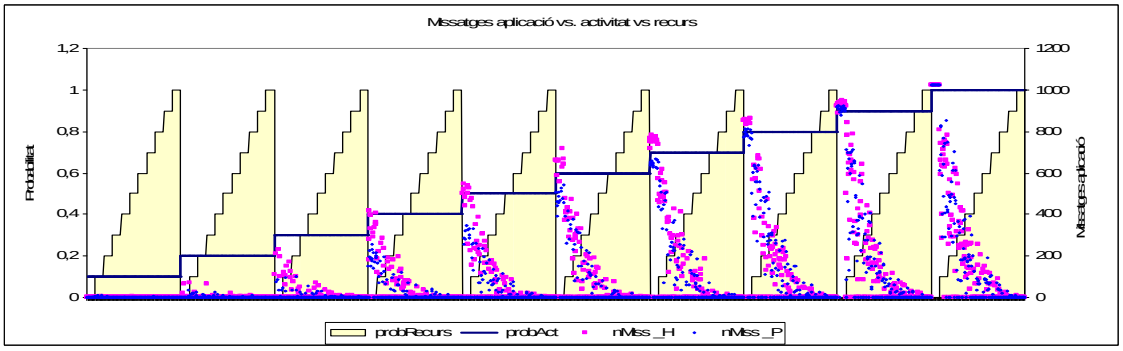


Fig. 5.19. Missatges d'aplicació per algoritme P i H, amb r=10.

- Fixant la dimensió de l'hipercub, el màxim de salts d'aplicació per al camí més llarg succeeix per probabilitats d'activitat de GIS mitjanes (entre 0,3 i 0,8) i en la franja baixa de la probabilitat de recurs conegut (entre 0 i 0,4). Això és degut a que amb aquestes condicions, es troben pocs camins disponibles i aquests han de ser el més llargs possibles.

- Com més gran és la divergència entre la probabilitat d'activitat de GIS i la probabilitat d'existència de recurs, major és el nombre de missatges

d'aplicació tramesos i el nombre de branques de cerca obertes, especialment per l'algoritme H.

- **No es pot assegurar la independència respecte de la topologia IP**. No obstant, es comprova com el canvi de la topologia IP i el fet de duplicar el nombre de routers de la xarxa no afecta al rendiment de l'algoritme de cerca en el nivell d'aplicació.

Com a conclusió global es pot extreure, a partir de les conclusions parcials anteriors, que l'algoritme H es comporta millor per a xarxes on els nodes GIS no tenen alts percentatges d'activitat, és a dir, que sovint entren i surten de la xarxa o que per diverses qüestions (col·lapse, excés de tasques, etc.) no poden atendre les peticions d'usuaris. En aquests casos, ja que l'algoritme H és capaç de trobar camins d'aplicació alternatius i intenta mitigar l'efecte dels nodes inactius, presenta un millor comportament i és capaç de trobar major nombre de recursos. No obstant, com a contrapartida, el cost en nombre de missatges d'aplicació i branques obertes (i per tant, en salts d'aplicació) augmenta.

Per altra banda, per a xarxes més estàtiques amb probabilitat d'activitat mitjanes, l'algoritme P es va assemblant cada vegada més, en termes de rendiment, a l'algoritme H. En aquests casos, els costs s'equiparen, fins arribar a coincidir quan la probabilitat d'activitat dels GIS és 1.

5.4. Segon Grup de Simulacions

5.4.1. Configuració

El segon grup de simulacions pretèn determinar la relació que implica el canvi de topologia a nivell IP al rendiment de l'algoritme de cerca. Més concretament, en les simulacions es manté el número de routers totals a 100 (o molt pròxim si la distribució no permet dividir el total de routers en el nombre desitjat d'AS), així com la dimensió de l'hipercub (dimensió 8), i el que es varia és l'agrupació dels routers en diferents tamanys d'AS i diferents graus de connectivitat.

Així doncs, les simulacions que es realitzen es resumeixen en la **Taula 5.6**:

Sim.	*Núm. AS*	*Intercon. entre AS*	*Núm. routers per AS*	*Intercon. entre routers*	*Dimensió hipercub*	*Probabilitat GIS actiu*	*Probabilitat recurs conegut*
1	10	4	10	6	8	0.1,0.2,...,1	0,0.1,...,1
2	8	4	12	6	8	0.1,0.2,...,1	0,0.1,...,1
3	6	4	17	6	8	0.1,0.2,...,1	0,0.1,...,1
4	5	3	20	6	8	0.1,0.2,...,1	0,0.1,...,1
5	4	2	25	6	8	0.1,0.2,...,1	0,0.1,...,1
6	2	1	50	6	8	0.1,0.2,...,1	0,0.1,...,1
7	2	1	50	20	8	0.1,0.2,...,1	0,0.1,...,1
8	2	1	50	10	8	0.1,0.2,...,1	0,0.1,...,1
9	2	1	50	30	8	0.1,0.2,...,1	0,0.1,...,1

Taula 5.6. Configuració de les simulacions del Grup2.

Les simulacions de 1 a 6 consisteixen en variar l'agrupació dels routers, conservant la connectivitat entre els mateixos. En canvi, en les simulacions 7, 8 i 9 es manté la mateixa distribució de routers per a cada AS i el que varia és la connectivitat dels routers de dins els AS.

En l'annex 8.15. es poden consultar les dades estadístiques obtingudes a partir de l'execució de les simulacions. A continuació es realitza el seu anàlisi.

5.4.2. Anàlisi de les Dades

En el segon grup de simulacions es comproven les mateixes característiques examinades pel primer grup. En aquest cas, les simulacions corroboren completament la independència dels algoritmes respecte de la topologia IP.

Consultant la **Taula 8.12** de l'annex 8.15.2. , es poden extreure una sèrie de conclusions:

- En missatges d'aplicació, el número màxim es manté constant a 256 (2^n), mentre que la mitjana per a totes les simulacions oscil·lant entorn de 23 missatges amb una variació de 2 o 3 missatges.

- Pel que fa al salts d'aplicació per a cada branca de cerca, el número màxim es correspon exactament amb el que indica cada algoritme per la dimensió especificada, mentre que en mitjana els resultats varien entorn de 1,5 branques aproximadament.

- Si s'examina el número de branques per cada cerca, el màxim també es manté constant (2^{n-1}) i la mitjana variant entorn a 12.

- En recursos trobats, el número màxim varia entorn de 50 i la mitjana entorn de 3,6 en totes les simulacions realitzades.

Per tots els punts anteriors, i veient que la variació de les dades de les mètriques a nivell d'aplicació obtingudes per a cada una de les simulacions varien poc entre elles, es pot assegurar que els algoritmes de cerca implementats no es veuen afectats per la topologia a nivell IP sobre la qual es munta el GRID.

No obstant, si que es pot observar com la tendència del nombre de salts de xarxa acumulats, tant les mitjanes com els màxims, decreix des de la primera simulació fins a la novena. Si es consulta la **Taula 5.6**, es pot veure que des de la simulació 1 fins a la 9, el que varia és l'agrupació en AS, disminuïnt el nombre d'AS i per tant, augmentant el número de routers per cada AS.

Així doncs, s'observa que com menys AS existeixen, la tendència és que s'acumulin menys salts de xarxa sota les mateixes condicions de simulació. Presumiblement, com que cada cerca necessita menys salts IP per executar-se completament, la latència global de la cerca també serà menor.

CAPÍTOL 6. BALANÇOS I CONCLUSIONS

6.1. Verificació dels Objectius

El propòsit clau del projecte, tal com s'explica a 1.2. ha sigut el d'implementar i evaluar dos algoritmes de cerca de recursos dins d'una arquitectura GRID organitzada en topologia hipercub. Es pot senyalar que aquest objectiu principal ha estat satifactòriament assolit, tant per la correcta implementació dels esmentats algoritmes com per l'anàlisi posterior a partir dels resultats de les simulacions realitzades.

La resta d'objectius imposats també han estat assolits, els quals es revisen tot seguit:

- Comprendre la topologia hipercub r-dimensional. Aquest objectiu es assolit en l'apartat 2.4. , on s'expliquen les característiques d'una topologia en hipercub. A més, mitjançant la implementació software, els coneixements adquirits s'han aplicat, reforçant la consecució de l'objectiu.

- Aprendre a usar el Tookit GridSim. En aquest cas, la pròpia implementació de l'aplicació usant GridSim, i la posterior extracció de conclusions a partir de les simulacions realitzades, fan que es demostri que aquest objectiu ha estat assolit.

- Dissenyar una aplicació que faci ús de les característiques de GridSim. A pesar d'algunes dificultats inherents de l'arquitectura i disseny de GridSim, s'ha aconseguit dissenyar una aplicació amb una forta orientació a objectes, que ha sabut adaptar els punts forts de GridSim, combinant-los amb decisions de disseny que han suposat un increment del rendiment en el desenvolupament de l'aplicació. El disseny inicial usant UML ha estat bàsic en la consecució d'aquest objectiu, i el seu èxit ha implicat un desenvolupament més planer evitant més complicacions.

- Desenvolupar un entorn on hi hagi una clara separació entre capa de xarxa i capa d'aplicació. En aquest cas, el propi disseny de l'aplicació ha fomentat la separació en dues capes de tota l'arquitectura a simular. Per una part, amb l'ús i adaptació de BRITE s'ha aconseguit generar topologies IP aproximades a la realitat, i per l'altra banda, el desenvolupament dels components del GRID envoltant els objectes simples de GridSim per a donar-lis majors funcionalitats, ha ajudat a aconseguir una topologia d'aplicació que fa ús de la topologia subjacent que té per sota. És a dir, s'ha aconseguit separar completament la capa IP de la capa d'aplicació, fins al punt que el procés de generació de la xarxa es pot usar en altres projectes on es variï la topologia d'aplicació o els nodes que la formen.

- Extreure estadístiques de les simulacions realitzades. Tal com es pot comprovar a les dades de sortida del simulador, i recopilades en format

de full de càlcul, s'han guardat múltiples mètriques que posteriorment han pogut ser analitzades tot extraïent mitjanes, valors màxim i mínims o la variabilitat de les mostres. A més, s'han pogut determinar relacions entre diferents mètriques i les dades segueixen disponibles per a posteriors anàlisis estadístics si s'escau.

- Generar topologies a nivell IP. Tal com s'indicava inicialment, era preferible buscar algun software que ajudàs o directament generés la topologia de nivell IP, ja que aquesta tasca es sortia dels objectius del projecte. Així doncs, es va trobar l'eina BRITE, la qual estant escrita en Java, incitava a la completa integració de les seves funcionalitats dins de l'aplicació del projecte, com així finalment s'ha fet. Per tant, es pot dir que aquest objectiu ha estat assolit de forma satisfactòria.

Encara que inicialment no formassin part dels objectius inicials, s'han assolit altres punts importants que poden ser usats en posteriors projectes i que han facilitat la rapidesa en les simulacions i les tasques de verificació de la implementació.

Primer de tot, cal destacar la decisió de disseny de separar la xarxa IP en subxarxes més petites (sistemes autònoms), amb menor número de routers per tal de facilitar la convergència de l'algoritme de routing, que es veia sobrepassat a l'hora de convergir en xarxes molt grans. D'aquesta forma, les xarxes han pogut creixer més, el temps de generació de la topologia usant l'algoritme d'enrutat ha disminuït dràsticament i la longitud de les taules d'encaminament també s'ha reduït. A més, relacionat amb la separació en subxarxes, s'ha hagut de modificar per complet l'algoritme RIP que implementava GridSim, sent aquest un dels punts més difícils de desenvolupar i, sobretot de fer debug, donada la gran quantitat de missatges i nodes implicats.

Cal destacar també el fet d'haver desenvolupat l'aplicació per tal que es pugui desplegar sobre un GRID real tot substituïnt GridSim i desenvolupant una petita capa d'adaptació entre el GRID físic i l'aplicació desenvolupada.

Un altre punt important ha estat el de trobar el visualitzador de topologies Otter, el qual ha ajudat a l'hora de realitzar les proves de funcionament dels algoritmes de cerca, ja que d'aquesta forma ha estat més fàcil comprovar els camins seguits per cada un dels missatges al llarg del GRID.

6.2. Balanç de les Eïnes Usades

Per començar, esmentar que l'ús de Java ha implicat un increment del rendiment en el desenvolupament de l'aplicació, ja que com sempre que s'usa, és fàcil trobar eines complementàries que ajuden en l'etapa de desenvolupament, com en aquest cas ha estat l'ús de BRITE i Otter. Per tant, llevat del fet que l'ús de Java venia implícit per la utilització de GridSim, ha estat la decisió correcte triar Java per implementar el codi, tant per l'extensa documentació disponible com per ser un llenguatge orientat a objectes amb tot el que això implica a l'hora de programar.

L'entorn de desenvolupament Eclipse ja era conegut i presenta una gran base d'usuaris, així com plugins de tercers que faciliten el desenvolupament. El plugin més important que s'ha usat és el de Subversion, per tal de versionar el codi font de l'aplicació i així no tenir problemes a l'hora de retornar a versions anteriors si alguna cosa falla, ja que Subversion ho fa automàticament.

Passant ja a les eines específiques del projecte, GridSim ha presentat algunes complicacions que sorgeixen a partir de la seva arquitectura i disseny. Entre altres, els problemes més importants que presenta són:

- Baixa escalabilitat. A mesura que el número de nodes (objectes) creats augmenta, la memòria RAM augmenta en una proporció més elevada.

- No implementar la interfície *serializable* en tots els objectes. El fet que les classes de GridSim no implementin la interfície de Java *serializable* ha impedit guardar en fitxer de forma fàcil tota la topologia, tant de xarxa com d'aplicació amb tots els seus objectes creats, per tal de recuperar-la posteriorment i així clonar la mateixa topologia entre diferents simulacions. La implementació de la interfície és tan fàcil com incloure en la definició de la classe "*implements Serializable*", sense haver d'implementar cap mètode. No coneixo el perquè d'aquesta decisió dels creadors, però algun motiu deuen tenir.

- L'arbre UML i per tant el disseny del Toolkit és, si més no baix el meu punt de vista, discutible. Aquesta afirmació sorgeix a partir de la separació dels objectes *Router,* i els seus derivats, en un branca diferent dels objectes *GridUser*, *AbstractGIS* i *GridResource*. Aquesta separació va complicar en el seu moment la creació de tots els objectes sota el paraigües de la interfície *NodeGrid* comentada a 4.4. Pretenia que tots els nous objectes extesos a partir dels objectes de GridSim es creassin i instansiassin de la mateixa forma, però la separació esmentada ho va impedir, havent de recórrer a realitzar alguns trucs basats en l'ús de Factorys internes.

- La implementació de RIP és errònia. Tal com ja s'ha comentat al llarg de la memòria, s'ha hagut de reimplementar l'algoritme RIP, ja que el que proporciona GridSim contenia errors importants. El més destacable és el de reenviar contínuament les taules d'encaminament sempre que un missatge RIP arribava al node, fet que dificultava enormement la convergència de l'algoritme. Actualment només es reenvien missatges si la taula d'encaminament ha canviat. Un altre punt a comentar és la impossibilitat d'implementar l'enviament de paquets de manteniment de la topologia cada 30 segons, tal i com especifica RFC de RIP, donat que no ha estat possible generar un fil d'execució dins del fil del propi objecte i coordinar-los correctament. No obstant, no és un punt massa important en el present projecte, ja que la topologia IP no varia al llarg de tota la simulació perque els routers no es desactiven en temps d'execució.

Per tot això, l'ús de GridSim no m'ha satisfet excessivament i a vegades ha sorgit la sensació d'obviar l'ús de GridSim i usar directament SimJava, el qual és la base de GridSim.

Deixant ja GridSim, BRITE ha estat de gran ajuda pel fet d'olvidar-se de la implementació de la generació de la topologia IP, així com l'evaluació de les seves característiques, fent-les més o menys fidedignes respecte a la realitat. Així doncs, només s'ha hagut d'integrar el motor de generació a l'aplicació del projecte. A més, el disseny de BRITE és molt correcte i facilita tant la seva extensió com el seu ús en terceres aplicacions com és el cas.

Per últim, Otter no està tan ben dissenyat com BRITE, fet que complicava la seva integració. Ja que no era un element clau en el projecte, Otter no s'ha integrat directament dins de l'aplicació, i només s'ha generat el fitxer d'entrada respectant el format d'Otter. No obstant, l'ús d'Otter ha estat molt còmode i ha estat de gran utilitat a l'hora de comprovar el funcionament del algoritmes implementats.

6.3. Conclusions Generals

Anteriorment s'ha verificat que els objectius inicials s'han assolit i inclús superat, resolent molts problemes que han sorgit al llarg del disseny i el propi desenvolupament de l'aplicació, expressant que la implementació ha estat satisfactòria.

Encara que en el capítol 5 s'exposen les conclusions detallades sobre el comportament dels algoritmes de cerca plantejats, es vol usar aquest apartat a mode de resum dels punts o conclusions més importants als que s'ha arribat.

S'ha pogut comprovar com els algoritmes de cerca, tal com es podia pensar inicialment, resten independents de la topologia subjacent sobre la qual es munta el GRID a nivell d'aplicació, variant sols el nombre de salts a nivell de xarxa. L'augment dels valors d'aquesta mètrica és evident a mesura que una xarxa va augmentat el seu diàmetre, ja que els paquets es veuen obligats a augmentar el nombre de salts que realitzen. A més, com menys interconnectada està la xarxa a nivell IP, és a dir, si el número d'enllaços entre routers és baix respecte del nombre total de routers, més salts IP es realitzen, ja que hi ha menys camins disponibles per arribar a un destí des del mateix origen. Aquests són uns resultats evidents i lògics, i que són palpables en totes les xarxes desplegades.

L'anàlisi de les taules de resultats mostren altres valors interessants, com la linealitat en el nombre de branques i el nombre màxim de missatges a nivell d'aplicació que es generen per un procés de cerca, respecte de la dimensió de l'hipercub. Així doncs, el número màxim de missatges tramesos en una xarxa de dimensió n està acotat a 2^n, mentre que per la seva part, el nombre de branques màximes obertes dins de l'hipercub de dimensió n està limitat a 2^{n-1}. Ambdós límits són idèntics pels dos algoritmes evaluats.

La mètrica que indica la correcte implementació dels algoritmes de cerca és la que mostra els salts de la branca de cerca més extensa. Tal com s'indica a la teoria, aquest paràmetre té relació amb la dimensió i el propi algoritme. Es comprova que fins a la dimensió 2, els dos algoritmes es comporten pràcticament igual, obtenint que es realitzen com a màxim 2 salts d'aplicació en tots dos casos. A partir de la dimensió 3, en canvi, es pot observar clarament com en l'algoritme P els salts màxims són *n*, mentre que en l'algoritme H són *n+1*.

Finalment, un resultat també important és l'obtingut evaluant a partir de quina dimensió, en mitjana, es pot assegurar que el recurs buscat es troba en el GRID. Es pot veure com a partir de la dimensió 5, en mitjana, el recurs es troba, i que en dimensions menors no es pot assegurar aquest extrem.

6.4. Millores i Línies Futures

Encara que s'hagin assolit tots els objectius inicials plantejats, sempre resten millores i futures tasques a desenvolupar en posteriors estudis i projectes.

D'entre les millores a l'actual aplicació, es poden destacar les següents:

- Millorar l'enrutat entre els AS. Actualment, per tal de no complicar encara més el desenvolupament de l'algoritme d'enrutat, el routing entre els sistemes autònoms és estàtic. Això implica que no tots els AS es coneguin entre sí, fet que pot implicar que a vegades el nombre de salts necessaris per arribar des d'un AS origen al de destinació sigui major que l'òptim. Aquest punt es pot millorar implementat un algoritme de routing com BGP o adaptar l'algoritme RIP per aquesta tasca.

- Millorar l'ús de memòria RAM. Un dels aspectes limitants a l'hora de simular és el tamany de la memòria RAM. Seria útil investigar el GridSim per determinar en quins punts del seu software es podria reduir l'ús de memòria.

Pel que fa a les línies futures, aquest projecte obri la porta a estudis paral·lels d'evaluació d'altres algoritmes de cerca de recursos en diverses topologies d'aplicació, tant hipercub o d'altres de diferents, i així comparar les seves característiques, propietats i el seu rendiment, per tal d'emetre conclusions de quines combinacions d'algoritmes de cerca i topologies són més adients. També es pot pensar en un possible desplegament de l'aplicació sobre un GRID real, aprofitant que sols seria necessari crear una capa d'adaptació entre les dues entitats.

Per una altra part, també es pot analitzar el futur de l'entorn en el que s'ha desenvolupat el projecte, és a dir, el GRID i el GridSim.

Pel que fa al GRID, s'espera que a mesura que la tecnologia vagi madurant i s'aconsegueixin nivells més elevats d'interoperabilitat, s'extengui el seu ús a més entorns dels actuals d'investigació, i s'afegeixin nous projectes que fins ara feien ús de la potència dels superordinadors. Aquesta extensió del GRID

dependrà també en gran mesura de l'amplada de banda disponible dels usuaris del mateix, ja que aquest és un element clau per l'èxit de l'arquitectura. Sembla que de cada vegada més s'ofereixen connexions a Internet amb més capacitat, però que encara no acaben de ser suficients per tal d'assolir els requeriments necessaris per segons quines tasques com la transmisió de grans quantitats de dades en el mínim temps.

Per altra banda, també és interessant evaluar el futur de GridSim. Sembla que els desenvolupadors del Toolkit segueixen ampliant les funcionalitats i millorant el paquet en cada nova versió que treuen a la llum, i per tant, dóna la sensació que el desenvolupament no cesarà en un curt termini de temps. No obstant, no acaben de resoldre problemes importants en el disseny i en l'arquitectura del Toolkit, com són l'excesiu ús de memòria, la insuficient documentació disponible (només es presenten una sèrie d'exemples bàsics i l'API parcialment documentada) i la dificultat d'extendre molts dels objectes que defineix. L'escalabilitat del Toolkit també s'hauria de millorar molt per ser millor simulador, ja que a mesura que s'afegeixen nous nodes, els requeriments de memòria RAM es fan inabastables. Per tot això, es conclou que actualment GridSim té molts problemes per executar simulacions amb milers de nodes i que no és idoni per aquestes simulacions grans.

6.5. Impacte Medioambiental

L'impacte directe sobre el mediambient del present projecte es limita al consum elèctric de les màquines que formen part del GRID, ja que no hi ha cost de desplegament d'infraestructures donat que s'utilitzen les existents en l'actualitat, i si més no, aquest seria un impacte indirecte.

Per altra banda, l'impacte del desenvolupament del projecte també es limita sols al consum elèctric dels ordinadors usats per a programar, simular i redactar la memòria.

6.6. Conclusions Personals

Personalment, aquest projecte m'ha servit sobretot per a dissenyar una aplicació en Java des de l'inici, donant molta importància al disseny UML previ. Pel que fa al desenvolupament, he aprofitat per aprofundir en els conceptes d'orientació a objectes, que he anat usant al llarg de tot el codi, així com l'ús de diversos patrons de disseny com per exemple factorys i commands. Per a fer-ho he tingut l'avantatge de no haver d'aprendre a programar en Java, ja que tenia certa base adquirida prèviament amb altres projectes acadèmics i laborals, ni de buscar i entendre els patrons de disseny, sinó que sols he hagut de localitzar on utilitzar-los i aplicar-los correctament.

A part d'això, la realització del projecte m'ha permès entrar en el món del GRID, coneixent els seus fonaments i les característiques del mateix, així com els camps d'aplicació. Al ser una xarxa de recursos distribuïts, molt semblant a una xarxa P2P, entra dins del meu camp d'interès, i per tant, per aquest costat he pogut aprofundir en arquitectures aplicables a xarxes P2P.

CAPÍTOL 7. REFERÈNCIES BIBLIOGRÀFIQUES

7.1. Procedents de Llibres

[1] Berman, F., Hey, A.J. i Fox, G.C., *GRID Computing*, John Wiley & Sons Ltd, Chichester, England (2003).

[2] Travostino, F., Mambreti, J. i Karmous-Edwards, G., *GRID Networks Enabling GRIDs with Advances Communication Technology*, John Wiley & Sons Ltd, Chichester, England (2006).

[3] Jacob, B., Brown, M., Fukui, K. i Trivedi, N., *Introducing To GRID Computing*, IBM Redbooks, Autins, Texas, USA (2005).

7.2. Procedents d'Articles i Papers

[4] Stockinger, H. "Defining de GRID: a snapshot of the actual view", Springer Science+Bussines Media, LLC, 1-15, Lausanne, Switzerland, (2007).

[5] Foster, I., Kesselman, C. i Tuecke, S. "The Anatomy of the GRID", 1-25 (any no especificat).

[6] Berstis, V., "Fundamentals of GRID Computing", *IBM Redbooks Paper*, 1-28 (2002).

[7] Medina, A., Lakhina, A., Matta, I. i Byers, J., "BRITE: Universal Topology Generator from a User's Perspective", Boston University, 1-47, (2001).

[8] Buyya, R. i Murshed, M., "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for GRID Computing", Monash University, Melbourne, Australia, 1-37, (any no especificat).

7.3. Procedents d'Internet

[9] http://gridcafe.web.cern.ch

[10] Pàgina web del projecte GridSim, http://www.gridbus.org/gridsim/

[11] Pàgina web de Simjava, http://www.dcs.ed.ac.uk/home/hase/simjava/

[12] Pàgina web de BRITE, http://www.cs.bu.edu/brite/

[13] Pàgina web d'Otter, http://www.caida.org/tools/visualization/otter/

CAPÍTOL 8. ANNEXES

8.1. Definicions

BRITE. Boston university Representation Internet Topology gEnerator.
GIS. GRID Information Service.
CAIDA. Cooperative Association for Internet Data Analysis.
GIMPS. Great Internet Mersenne Prime Search.
AS. Sistema Autònom.
BGP. Border Gateway Protocol.
RIP. Routing Information Protocol.

8.2. Estàndards i Especificacions Aplicables al GRID

El següent llistat enumera un conjunt d'estàndards, recomanacions i especificacions relacionats amb l'arquitectura GRID:

- **OGSA** (Open GRID Service Architecture). Publicat pel GGF (Global GRID Forum), defineix l'especificació del nucli d'un GRID per tal d'aconseguir la interoperabilitat entre diferents implementacions i solucions. *(Referència: http://www.ggf.org).*

- **OGSI** (Open GRID Service Interface). És una extensió de l'especificació OGSA que defineix els mecanismes de creació, administració i intercanvi d'informació entre els serveis que proporciona el GRID. OGSI defineix el llenguatge WSDL (Web Service Definition Language) per tal de descriure les funcionalitats i forma d'accés al serveis del GRID. *(Referència: http://www.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf).*

- **OGSA-DAI** (OGSA- Data Access and Integration). S'ocupa de definir la forma d'accés i integració de dades des de fonts diferents a través del GRID. *(Referència: http://www.ogsadai.org.uk/).*

- **GridFTP**. És un protocol de transferència de dades segur i fiable, basat en el protocol FTP, que proporciona alt rendiment en les transferències i està optimitzat per xarxes d'àrea extensa amb grans amplades de banda disponibles. *(Referència: http://www.globus.org/GRID_software/data/gridftp.php).*

- **WSRF** (Web Services Resource Framework). Són una sèrie d'especificacions, remeses a OASIS per a convertir-se en estàndar, que defineixen la relació entre els serveis web i els recursos exposats a través del GRID. És el conjunt d'especificacions que apareixen com a resposta a la falta de coordinació entre els diferents actors encarregats d'estandaritzar i implementar arquitectures GRID. Recentment, la comunitat GRID i la de Web Services han unit esforços i a partir d'aquí apareix WSRF. La **Fig. 8.20** ho mostra de manera gràfica. *(Referència:*

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf i http://www.globus.org/grid_software/ecology.php). Les principials característiques de WSRF i l'especificació WS-Notification, directament relacionada amb la primera són les següents:

- Noms i vincles. Cada element del GRID té un nom que l'identifica unívocament i presenta un o més serveis per a interactuar-hi.
- Cicle de vida. És la base de la gestió front les fallades.
- Model d'informació. Les propietats dels recursos estan associades als mateixos recursos. Existeixen procediments per a consultar i establir les propietats i notificacions asíncrones per a notificar els seus canvis.
- Grups de serveis. Són bàsics per a serveis col·lectius i poden ser gestionats tant a nivell global com a nivell de membres individuals.

- **Estàndards relacionats amb Web Services**. Ja que els serveis exposats en el GRID estàn molt relacionats amb Web Services, també els estàndards relacionats són usats en l'arquitectura GRID a través de WSRF. Entre ells, es poden destacar XML, WSDL, SOAP i UDDI. *(Referència: http://www.w3.org/2002/ws/).*

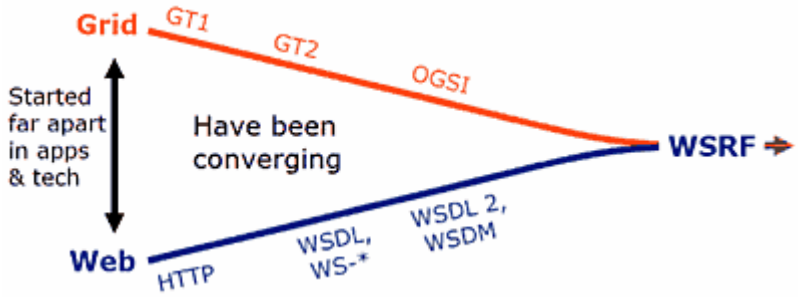


Fig. 8.20. Convergència de GRID i Web Services en WSRF.

8.3. Exemples de GRID

Alguns exemples de projectes que es fonamenten en l'ús de l'arquitectura GRID per a obtenir recursos de còmput es llisten acontinuació:

- **SETI@home**. És un projecte sorgit al Laboratori de Ciències de l'Espai de la Universitat de California i que pretèn cercar per possibles evidències de transmissions de radiofreqüència per formes de vida intel·ligents extraterrestres usant les dades capturades pels radiotelescopis de Arecibo, Puerto Rico. Les dades captades són digitalitzades i trossejades en temps i freqüència, i seguidament distribuïdes al llarg dels ordinadors personals que tenen instal·lada l'aplicació d'anàlisi. L'anàlisi es basa en buscar pics freqüencials o

temporals, i senyals Gaussians que poden ser produïts artificialment per algun tipus de sistema de transmissió. Una vegada s'ha analitzat, es retornen els resultats. Actualment, a març de 2008, la capacitat de càlcul del GRID ascendeix fins als 444Tflops aproximadament *(Referència: http://boincstats.com/stats/project_graph.php?pr=sah).*

- **Folding@home**. És mantingut per la Universitat d'Stanford, i el seu objectiu és el d'entendre els processos de formació de proteïnes, és a dir, com es combinen les mol·lècules entre sí mateixes per a donar lloc a estructures funcionals. Aquest és un problema fonalmental en l'estudi de la biologia mol·lecular i que contribueix a entendre i intentar trobar les causes i les possibles solucions a malalties com l'alzeimer, parkinson o el càncer. A setembre de 2007, la capacitat de càlcul del GRID és d'aproximadament 1224 Tflops. *(Referència: http://fah-web.stanford.edu/cgi-bin/main.py?qtype=osstats).*

- **GIMPS** (Great Internet Mersenne Prime Search). Es tracta d'un projecte que té per objectiu la recerca dels anomenats nombres primers de Mersenne. Matemàticament, un número Mersenne és aquell nombre que és una unitat menor que una potència de dos, això és 2^n-1. Per tant, un nombre primer de Mersenne, és un primer que compleix aquesta condició. Actualment, el nombre més gran trobat que compleix les condicions té un total de 9808358 dígits. A febrer de 2008, la capacitat computacional distribuïda del projecte ascendeix fins a uns 27Tflops. *(Referència: http://mersenne.org/primenet/).*

8.4. Pseudocodis i Informació Extesa dels Algoritmes de Cerca

En aquest annex es poden consultar els pseudocodis del funcionament dels dos algoritmes de cerca tractats al llarg del projecte, així com l'explicació detallada de cada una de les passes amb les que es divideix la seva execució.

8.4.1. Algoritme P

La **Fig. 8.21** mostra el pseudocodi corresponent a l'algoritme P.

```
1  satisfyRequest = processRequest(message.request);
2  IF (NOT satisfyRequest) THEN
3      IF (startNode) THEN
4          v_d = {0, 1, ..., n−1};
5      ELSE
6          v_d = message.v_d;
7      ENDIF
8      v_d, nnon-alive = statusNeighbors(v_d );
9      FOR k=0 TO (v_d.size()- nnon-alive – 1) DO
10         message.v_d = createList(k, v_d);
11         sendToNeighbor(v_d [k], message);
12
```

```
13      ENDFOR
    ENDIF
```

Fig. 8.21. Pseudocodi algoritme P.

Seguidament s'enumeren de forma detallada els passos a seguir i que expliquen perfectament el funcionament de l'algoritme P.

1. Quan un missatge de petició de servei és rebut per un node es crida a la funció *processRequest()*. Si la petició inclosa en el missatge no pot ser satisfeta, el node estableix la variable *satisfyRequest* a fals i s'inicia la propagació de la petició. En cas contrari, *satisfyRequest* agafa el valor vertader i no es produeix reenviament. El missatge es composa de la petició (*message.request*) i la llista de dimensions (*message.*v_d).

2. Si el node que reb la petició no la pot satisfer i és el node inicial del procés de cerca, la llista v_d s'inicialitza a v_d = *{0, 1, ..., n-1}* (la llista completa de dimensions). En altre cas, la llista v_d s'inicialitza amb la que arriba amb el missatge. En ambdós casos, la llista conté el conjunt de dimensions (veïns) a les quals el missatge ha de ser propagat. *Nota*: si es tracta del node inicial, ha de reenviar a tots els seus veïns.

3. El node crida a la funció *statusNeighbors(*v_d*)* i reordena la llista v_d, de manera que les dimensions corresponents als nodes que no estan actius es posicionen a les últimes posicions de la llista. Remarcar que es considera que un node és no actiu quan per qualsevol circumstància (càrrega elevada, tall de l'enllaç, caiguda, etc.) és innaccesible des de la xarxa. La funció esmentada, a més, retorna un valor enter que indica el número de node no actius en la llista reordenada.

4. Per cada posició *k* en la llista v_d que representa un node veï actiu, l'algoritme crida a la funció *createList(k,*v_d*)*, la qual crea una nova llista que conté les dimensions immediatament posteriors a la posició a la que apunta *k*. Per a cada node actiu veï, la llista *message.*v_d és inicialitzada. La llista i la petició de recurs original s'envia al node veï corresponent a la dimensió v_d*[k]* mitjançant la funció *sendToNeighbord()*.

8.4.2. Algoritme H

La **Fig. 8.22** mostra el pseudocodi corresponent a l'algoritme H.

```
1  satisfyRequest = processRequest(message.request);
2  IF (NOT satisfyRequest) THEN
3      IF (startNode) THEN
4          v_d = {0, 1, ..., n−1};
5          v_a = { };
6      ELSE
7          v_d = message.v_d;
8          v_a = message.v_a;
```

```
9     ENDIF
10    v_d, nnon-alive , dalive = statusNeighbors(v_d );
11    IF (nnon-alive > 1) THEN
12       v_a2 = addToList(v_a, dalive );
13    ENDIF
14    FOR k=0 TO (v_d.size()- nnon-alive – 1) DO
15       message.v_d = createList(k, v_d);
16       IF (v_d [k] = dalive ) THEN
17          message.v_a = v_a2;
18       ELSE
19          message.v_a = v_a;
20       ENDIF
21       sendToNeighbor([v_d [k], message);
22    ENDFOR
23    FOR j=0 TO (v_a.size() – 1) DO
24       IF (neighbor v_a [j] is not the parent node)
25          message.v_d = { };
26          message.v_a = { };
27          sendToNeighbor([v_a [j], message);
28       ENDIF
29    ENDFOR
30 ENDIF
```

Fig. 8.22. Pseudocodi algoritme H.

Els següents sis punts detallen l'execució de l'algoritme H.

1. Quan un nou missatge es rebut, es crida la funció *processRequest(message.request)*. Si la petició es pot satisfer, la variable *satisfyRequest* val *true* i no es produeix cap propagació de missatges. En cas contrari, *satisfyRequest* pren el valor fals i el missatge es reenvia als nodes veïns, juntament amb dues llistes: $message.v_d$ i $message.v_a$.

2. Si la petició no es pot satisfer i el node que reb la petició és l'inicial, la llista vd s'inicialitza a v_d=*{0, 1, 2, ..., n-1}* (la llista completa de dimensions) i v_a, per la seva part, s'inicialitza amb una llista buida, és a dir, v_a=*{ }*. En altre cas, les llistes s'inicialitzen amb els valors del missatge d'entrada. En ambdós casos, les llistes representen el conjunt de dimensions a través de les quals s'ha de propagar el missatge.

3. El node crida a la funció *statusNeighbors(v_d)* i reordena la llista v_d de manera que les dimensions corresponents a nodes veïns no actius es col·loquen en les darreres posicions de la llista. A més, la funció retorna dos valors enters: *numNonAlive* i *dAlive*. El primer enter representa el número de nodes no actius de la llista v_d, mentre que el segon mostra la dimensió del darrer node actiu de la llista v_d.

4. Si el nombre de nodes no actius es major que 1, es crida a la funció *addToList(v_a, dAlive)* que s'encarrega d'afegir *dAlive* a la llista v_a.

5. Per cada posició *k* de la llista v_d, es crida a la funció *createList(k, v_d)*, la qual crea una nova llista a partir de v_d que conté totes les dimensions guardades després de la posició *k* de la llista v_d. A més, la llista v_a és inicialitzada. Si la posició *k* de v_d es correspon amb *dAlive*, la llista v_a s'inicialitza a la llista retornada per la passa 4. En altre cas, v_a pren els valors de v_a rebut des del node veï. La petició i les dues llistes creades s'envien als nodes corresponents a la dimensió $v_d[k]$ a través de la funció *sendToNeighbor($v_d[k]$,message)*.

6. Finalment, el node intenta propagar la petició a cada un dels nodes de les dimensions guardades en v_a, només si el corresponent veï no és el node pare. El missatge està composat per la petició i les llistes v_d i v_a buides.

8.5. Instal·lació i Configuració de l'Entorn

8.5.1. Java

La instal·lació de Java es molt similar tant si es realitza en Windows com en Linux, l'únic que varia és l'inici de la mateixa i la configuració posterior. A continuació es mostren les passes a seguir per a instal·lar i configurar el JDK en els dos sistemes operatius.

Linux

L'arxiu que es descarrega és un executable binari amb extensió *.bin*. El procés d'instal·lació és el següent:

1. Donar-li permís d'execució

```
chmod +x jdk-<versio>-linux-i586.bin
```

2. Moure's al directori on es vulgui instal·lar, per exemple

```
cd /usr/local/ (es necessiten permisos de root en aquet cas)
```

3. Executar l'arxiu. Es crear la carpeta jdk-<versio>-linux-i586/

```
./jdk-<versio>-linux-i586.bin
```

4. Comprovar que la versió utilitzada de Java és la de Sun. La comanda demana seleccionar la versió Java desitjada.

```
update-alternatives –config java
update-alternatives –config javac
```

5. Crear un enllaç simbòlic per facilitar la migració entre versions Java

```
ln -s /usr/local/jdk-<versio> /usr/local/java
```

6. Establir la variable d'entorn JAVA_HOME

```
export JAVA_HOME=/usr/local/java (executar amb l'usuari no root)
```

7. Afegir els executables de Java a la variable d'entorn PATH per tal de tenir un accés més fàcil a aquestes eines.

```
nano ~/.bashrc
afegir al final de l'arxiu
export PATH=$PATH:/usr/local/java/bin
```

8. Comprar que tot és correcte, fent java -version. Si apareixen unes línies similars a les següents, la instal·lació ha estat satisfactòria:

```
java version "1.6.0_02"
Java(TM) SE Runtime Environment (build 1.6.0_02-b05)
Java HotSpot(TM) Client VM (build 1.6.0_02-b05, mixed mode, sharing)
```

Windows

Executar l'arxiu .exe i seguir les instruccions.

Nota: amb Ubuntu i possiblement amb altres distribucions, també es poden trobar empaquetades versions de Sun en repositoris Multiverse o non-free. En aquest cas, sols fa falta executar com a root:

```
aptitude install sun-java6-jdk
```

8.5.2. GridSim

Requeriments

A continuació es detallen els requeriments necessaris per a usar el toolkit així com instruccions d'instal·lació del mateix.

L'únic requeriment necessari per a desenvolupar i simular amb GridSim és el de disposar de l'entorn Sun JDK (Java Developer Kit) 1.4.2 o més actual. Versions Java externes de les de Sun pot ser que no siguin compatibles amb GridSim, com per exemple gcj o J++.

Al ser un entorn de desenvolupament íntegrament Java, és independent del sistema operatiu que s'utilitzi. En aquest cas, el sistema operatiu usat és Linux.

En cas que es vulgui compilar les fonts Java de GridSim, cal tenir instal·lat i configurat Ant.

Instal·lació

Per tal d'instal·lar el toolkit sols és necessari descomprimir l'arxiu descarregat en algun directori del sistema.

Per a comprovar que tot l'entorn de treball s'ha instal·lat i configurat correctament, el més indicat és intentar compilar i executar un exemple dels que porta el paquet GridSim per defecte.

1. Compilar l'arxiu de fonts Java

```
En Windows: javac -classpath %GRIDSIM%\jars\gridsim.jar:. Example1.java
En Linux: java -classpath $GRIDSIM/jars/gridsim.jar:. Example1.java
```

2. Executar l'arxiu de classe Java

```
En Windows: java -classpath %GRIDSIM%\jars\gridsim.jar:. Example1
En Linux: java -classpath $GRIDSIM/jars/gridsim.jar:. Example1
```

Tant la clau $GRIDSIM (Linux) com %GRIDSIM% (Windows) es refereixen a la ruta on està instal·lat GridSim dins del sistema operatiu.

Per tal d'augmentar la memòria disponible per a la màquina virtual Java en simulacions llargues, es poden executar les classes generades amb la següent comanda:

```
java -Xmx300m -classpath $GRIDSIM/jars/gridsim.jar:. Example1
```

En aquest cas, 300 indica que la màxima memòria usable per a la màquina virtual Java són 300MB.

8.5.3. BRITE

Requeriments

Tant si s'usa la versió Java com la C++ en mode interfície gràfica, s'ha de tenir correctament instal·lat i configurat l'entorn d'execució de Java, precissant com a mínim JDK1.3. En cas d'usar-se C++ en mode sols de línies de comandes, no és necessari cap requeriment afegit.

Instal·lació

La instal·lació de BRITE en local està composada de les següents accions:

1. Descomprimir el fitxer descarregat

```
tar -xzvf BRITE.tar.gz
```

2. Dirigir-se al directori, per defecte BRITE

```
cd BRITE
```

3. Compilar les fonts

```
make all
```

4. Donar permisos d'execució a l'script de llançament de BRITE

```
chmod +x brite
```

Una vegada instal·lat, la interfície gràfica es llança executant el fitxer brite:

```
./brite
```

El contingut d'aquest fixer executable és simplement el següent, on es pot veure que crida a la màquina virtual Java, passant-li per paràmetre el tamany de memòria reservat (-Xmx256M), el directori de classes (-classpath Java/:.) i la classe llançadora de l'aplicació (GUI.Brite)

```
#!/bin/sh

java -Xmx256M -classpath Java/:. GUI.Brite
```

Notar que si s'ha de generar una topologia de xarxa complicada, amb gran quantitat de nodes, pot ser d'utilitat augmentar la memòria reservada, modificant el seu tamany amb el paràmetre –Xmx.

8.5.4. Otter

Requeriments

L'únic requeriment és el de disposar de l'entorn d'execució Java, com a mínim la versió JDK 1.2.

Instal·lació

La instal·lació requereix la compilació de les fonts d'Otter. El propi arxiu descarregat porta inclòs un arxiu Makefile per tal de procedir fàcilment a la compilació de l'aplicació. Així doncs, el procés de compilació i instal·lació és el següent:

1. Descomprimir l'arxiu baixat:

```
tar -xzvf otter-[versio].tar.gz
```

2. Dirigir-se al directori, per defecte otter:

```
cd otter
```

3. Compilar les fonts amb la utilitat make

```
make
```

4. Executar:

```
./otter
```

8.5.5. Subversion

Subversion és un software de sistema de control de versions. Es tracta de software lliure, llicenciat baix una llicència de tipus Apache/BSD i també se'l conèix com svn, per ser el nom de l'eina de línia de comandes.

Durant el projecte, Subversion s'ha encarregat del control de versions de l'aplicació, mitjançant l'ús del plugin Subeclipse per l'IDE Eclipse.

8.5.6. Eclipse

Eclipse és un entorn integrat de desenvolupament, de codi obert, implementat en Java i que originariament s'usava per a programar aplicacions en aquest llenguatge. Actualment, Eclipse s'usa com a plataforma de desenvolupament per a múltiples llenguatges de programació donada la seva modularitat i la capacitat d'incloure infinitat de plugins que integren més funcionalitats afegides.

8.5.7. Full de Càlcul

De tots és conegut el que és un full de càlcul. Resumint, és una aplicació de manipulació de dades, bàsicament numèriques. El software més popular de full de càlcul és l'Excel de Microsoft Office, el qual s'ha usat per a generar les gràfiques, així com també OpenOffice Calc, l'alternativa lliure a l'Excel, desenvolupada en Java, que s'ha usat per volcar les dades procedents dels fitxers de text separats per tabuladors csv.

8.5.8. Altres

Altres eines secundàries usades són les següents:

- **SSH**. Aplicació i protocol per accedir de forma segura a màquines remotes. Proporciona autenticació i encriptació de les dades intercanviades entre les màquines. S'ha usat per accedir a l'ordinador remot sobre el qual s'han realitzat les simulacions.

- **Kate**. Editor de text gràfic, que pertany a l'escriptori KDE. Usat per a modificar i visualitzar el contingut de fitxers de text, principalment scripts.

- **Nano**. Editor de text en mode consola. Usat per a modificar i visualitzar fitxers de text, bàsicament scripts, en conjunció amb SSH.

8.6. Arbres UML de BRITE i GridSim

L'arbre UML de BRITE és el que s'indica a la **Fig. 8.23**:

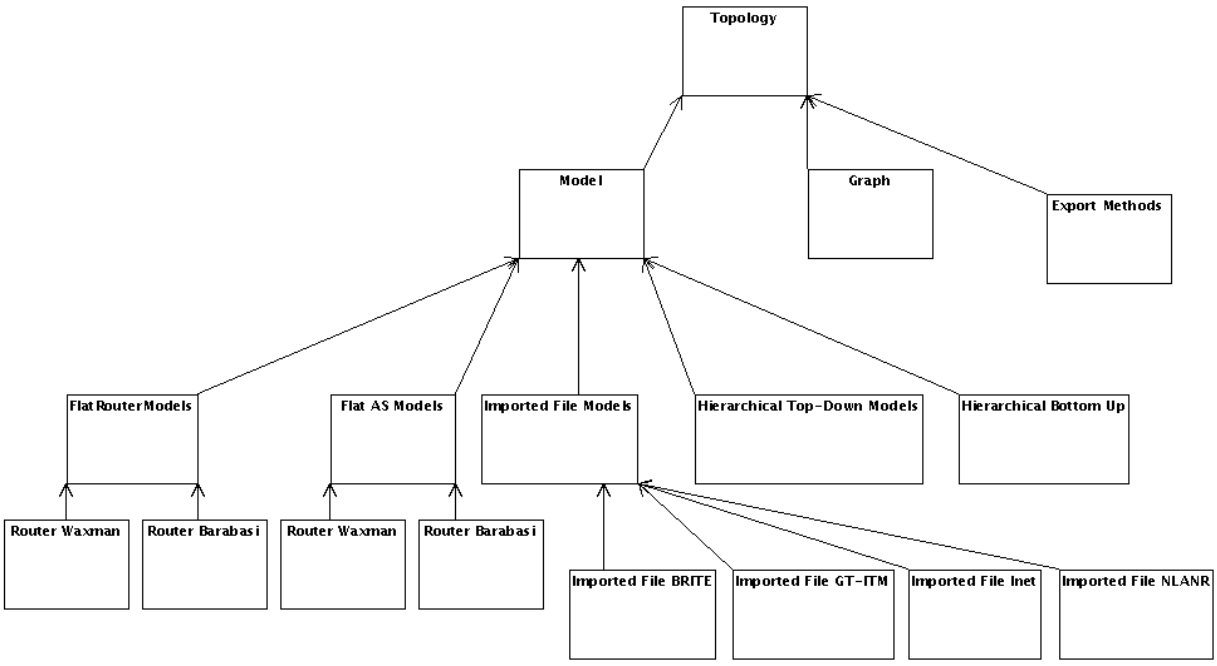


Fig. 8.23. Arbre UML de BRITE.

L'arbre UML de GridSim és el que es mostra a **Fig. 8.24**.

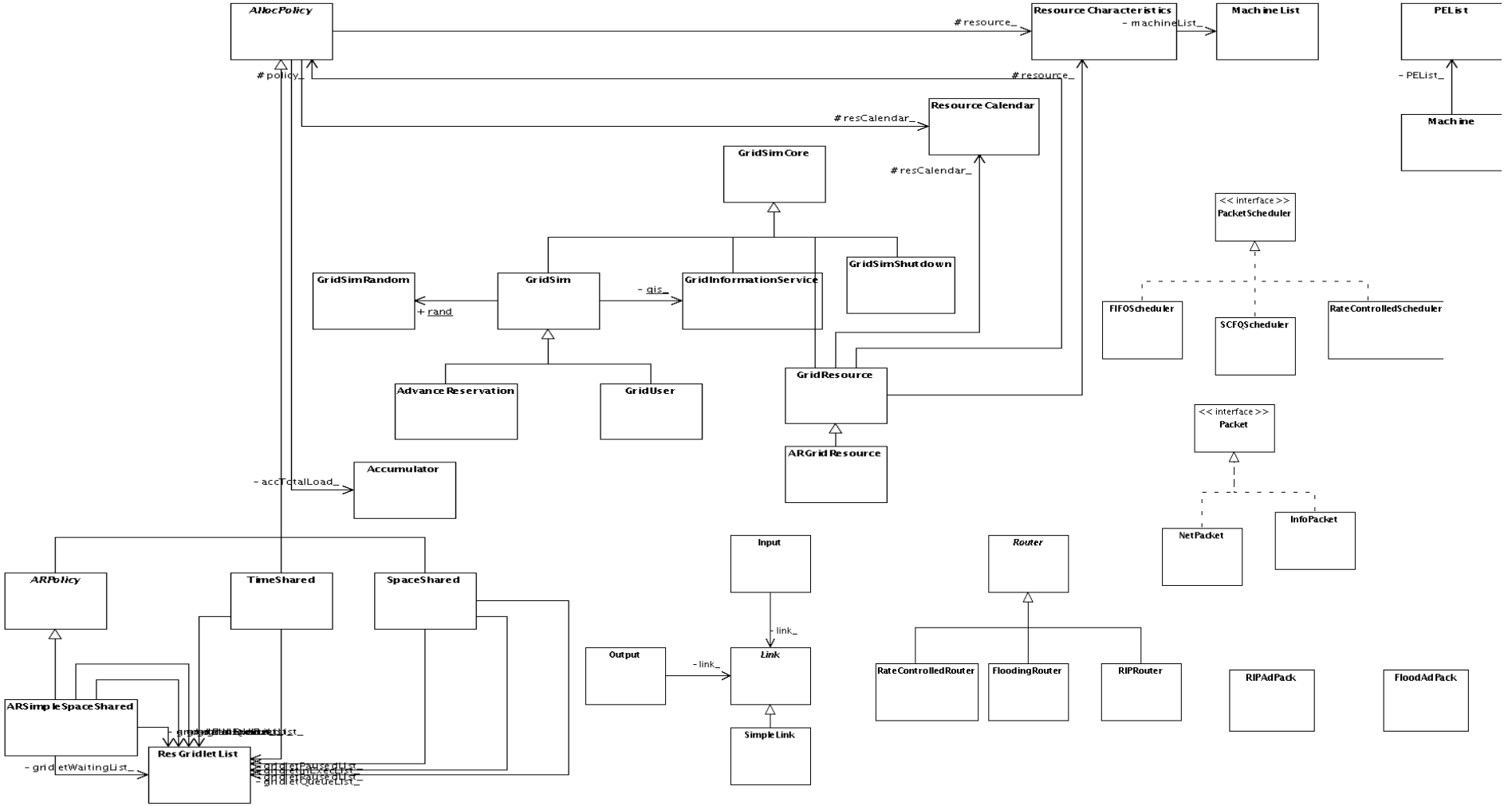


Fig. 8.24. Arbre UML de GridSim.

8.7. Diagrames UML de l'Aplicació

8.7.1. Diagrama Global

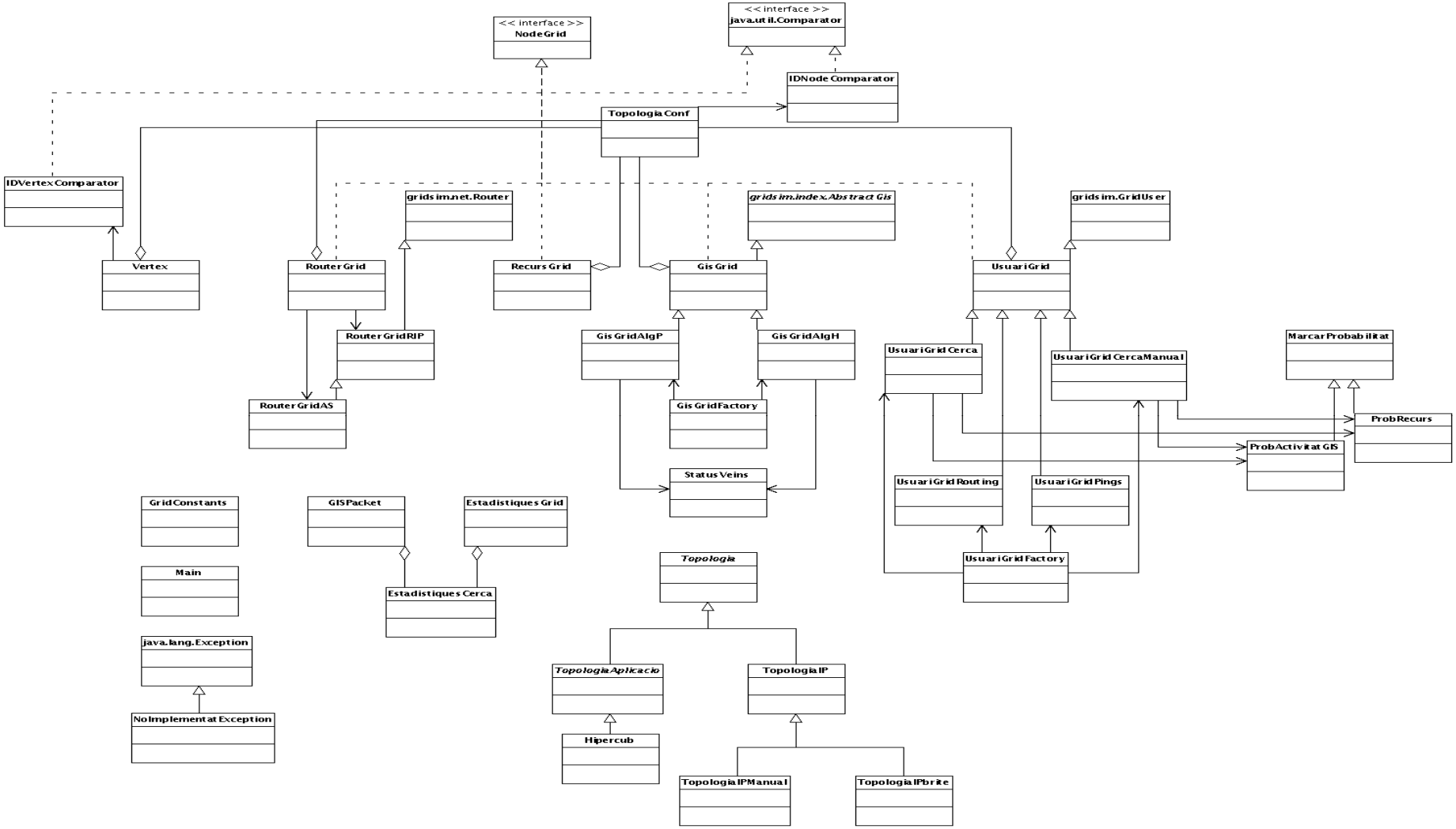


Fig. 8.25. Diagrama UML de l'Aplicació.

8.8. Diagrames de Fluxe de l'Aplicació

8.8.1. Diagrama UsuariGridCerca

La **Fig. 8.26** exemplifica el diagrama de fluxe de la classe UsuariGridCerca, mitjançant la qual es controla tot el procés de llançament de processos de cerca i recolecció de dades estadístiques. Les seva funcionalitat és recòrrer totes les combinacions de probabilitats d'activitat de GIS i presència de recurs, simulant un procés de cerca amb aquests paràmetres i repetir-lo 20 vegades per extreure resultats mitjans amb una certa base de valors.

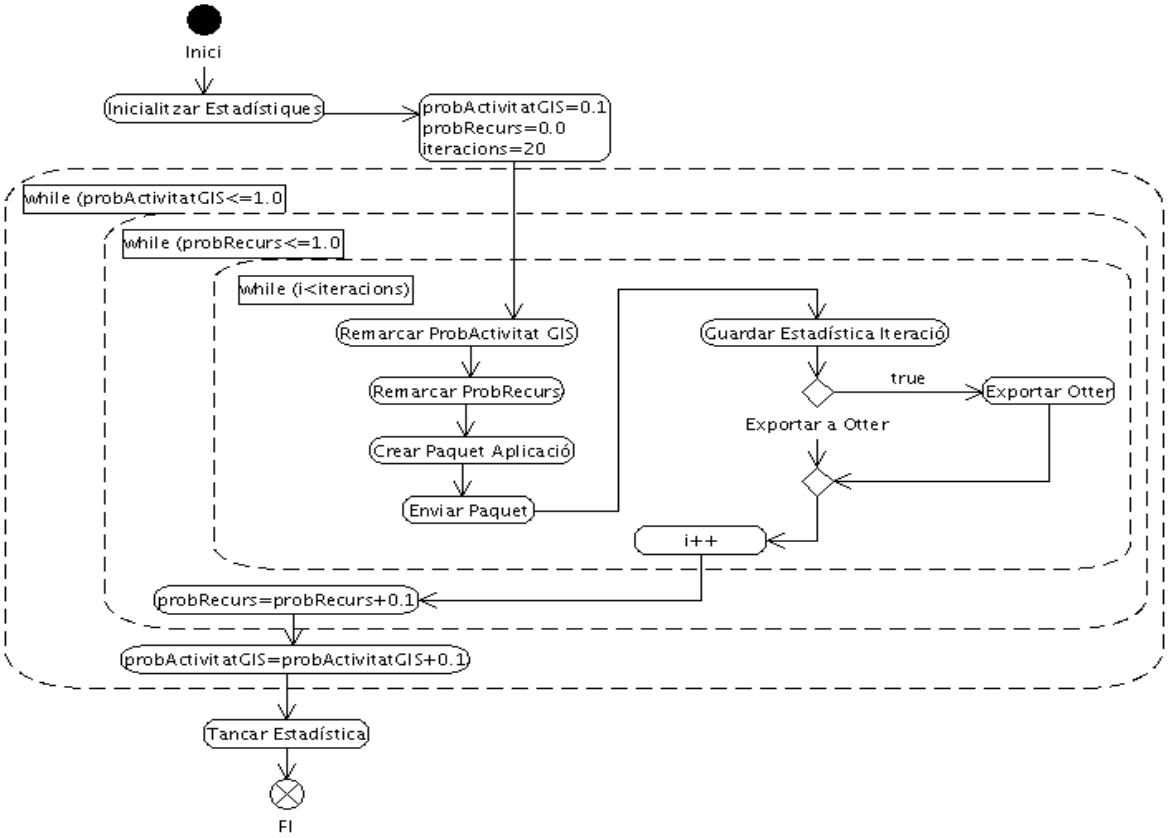


Fig. 8.26. Diagrama de Fluxe de la Classe *UsuariGridCerca*.

8.8.2. Diagrama de RouterGridRIP i RouterGridAS

El diagrama de la **Fig. 8.27** mostra les accions que implementen els routers desenvolupats, en aquest cas, el router RIP intern de cada AS i els routers frontera. Es distingeixen dos processos depenent del paquet entrant: enrutat d'un paquet (que pot ser un paquet de ping o un paquet de nivell d'aplicació, *GISPacket*) i processament d'un anunciament de routing.

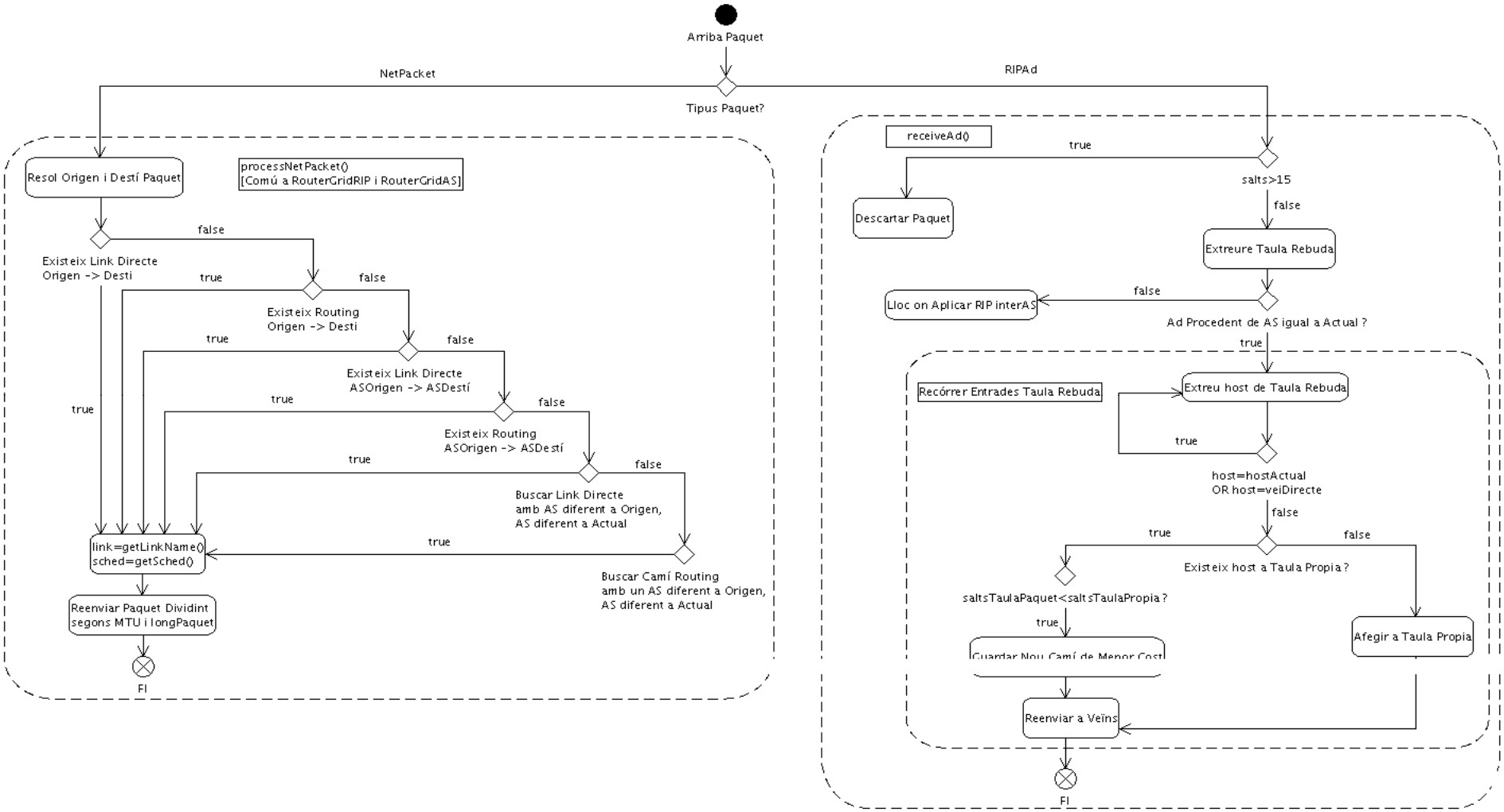


Fig. 8.27. Diagrama de fluxe de les accions realitzades quan arriba un paquet a *RouterGridRIP* i *RouterGridAS*.

8.8.3. Diagrama Global

El diagrama de la **Fig. 8.28** indica el fluxe de l'aplicació a nivell global. En ella es pot veure l'encadenament d'accions executades des de que s'arranca l'aplicació fins que finalitzen les simulacions.

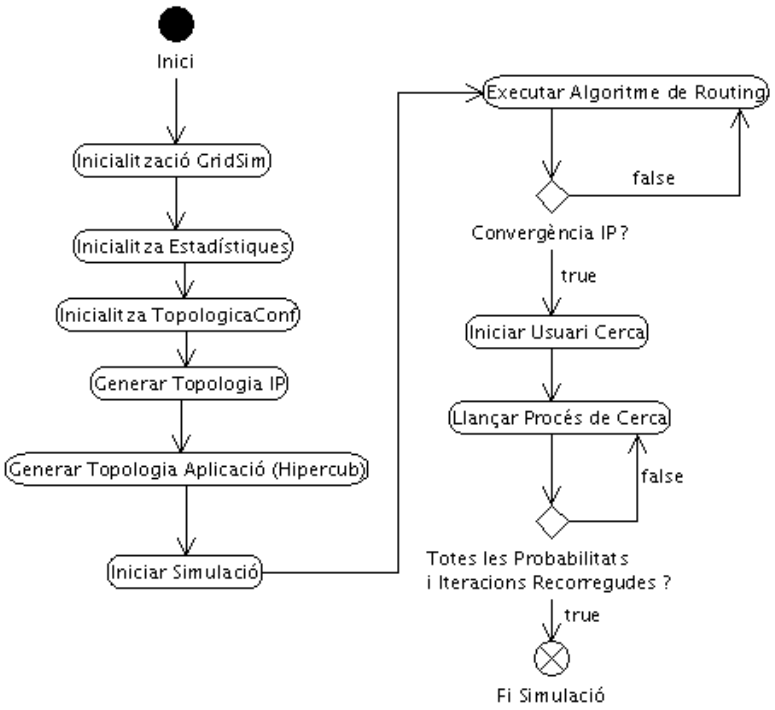


Fig. 8.28. Diagrama de fluxe de l'aplicació.

8.9. Format Fitxer Configuració BRITE

El fitxer usat per a configurar el motor de generació de topologies BRITE en el present projecte (**Fig. 8.29**), està dividit en quatre seccions diferenciades. L'arxiu està preparat per a generar un topologia dividida en sistemes autònoms, i per aquesta raó es defineixen quatre parts.

La primera configura l'entorn macroscòpic de la topologia, és a dir, les amplades de banda disponibles entre sistemes autònoms. La segona realitza la configuració pròpia dels sistemes autònoms. La tercera s'encarrega de definir els paràmetres de xarxa dels nodes localitzats dins dels sistemes autònoms. Finalment, la quarta indica en quins tipus de fitxers s'ha d'exportar la topologia generada.

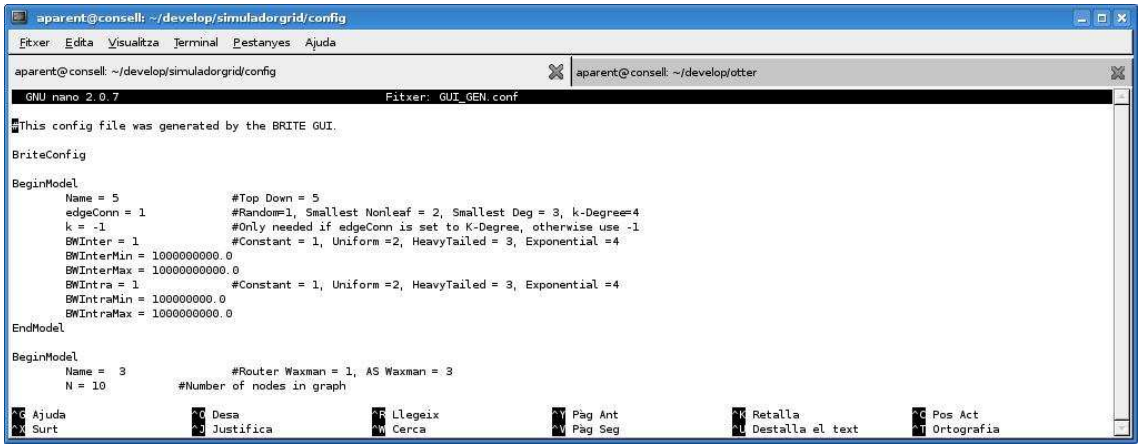


Fig. 8.29. Captura de l'arxiu de configuració de BRITE.

Per a majors referències, es pot consultar el manual de BRITE: http://www.cs.bu.edu/brite/.

8.10. Script d'Execució de l'Aplicació

L'script bash té el següent aspecte (**Fig. 8.30**):

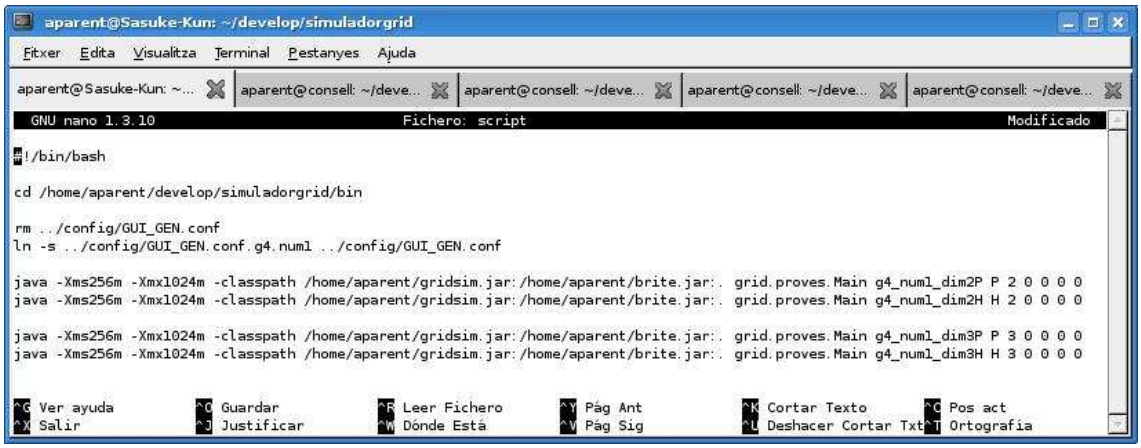


Fig. 8.30. Captura de l'script d'execució de l'aplicació.

Les comandes "*rm ../config/GUI_GEN.conf*" i "*ln –s ../config/GU_GEN.conf.g4.num1 ../config/GUI_GEN.conf*" s'encarreguen de configurar l'arxiu d'entrada de BRITE. Més concretament, la primera comanda elimina, si existeix, l'enllaç simbòlic GUI_GEN.conf. Aquest arxiu és el que busca l'aplicació per a carregar els valors de configuració de BRITE. Llavors, la segona comanda crea un nou enllaç simbòlic, amb el mateix nom, però que en aquest cas apunta a l'arxiu GUI_GEN.conf.g4.num1.

Per la seva banda, les següents línies es corresponen amb la comanda que executa l'aplicació Java amb els paràmetres de configuració del seu comportament que es desitgin, els quals es poden consultar a 8.17.

8.11. Format Fitxer d'Entrada a Otter

El fitxer de text que serveix d'entrada de dades a Otter presenta un format basat en l'ús de tags a l'inici de cada línia del fitxer, indicant la informació que representa i el tipus de dada de la que es tracta.

Així doncs, els tags usats són els detallats a continuació (**Taula 8.7**).

Tag	*Significat*	*Format*	*Paràmetres*
Data	Data de les dades del fitxer (Opcional)	d any mes dia	any: integer mes: integer (0-11) dia: integer
Tamany de gràfic	Número de nodes (Ha d'apareix abans del llistat de nodes)	t número	número: integer
	Número d'enllaços (Ha d'apareix abans del llistat de nodes)	T número	número: integer
	Tamany dels objectes	s tamany	número: integer
Nodes	Propietats dels nodes	n index_node x y nom	index_node: integer identificador únic x: integer positiu posició x en el mapa y: integer positiu posició y en el mapa nom: string no espais nom del node lat: float latitud que ocupa long: float longitud que ocupa
		N index_node lat long nom	
		? index_node nom	
Grups	Inicialització d'un grup	g index s|d num descripció	index: integer identificador únic s: char grup conté doubles d: char grup conté strings num: integer

			número d'entrades descripció: string descripció del grup
Casella	Propietats de cada grup (Ha d'apareixer després del tag Grups)	f index_grup string'string2'...	index_grup: integer identificador del grup al qual s'aplica la propietat string*: string valors de les propietats separats per '. El número d'strings que apareixen ha de ser igual a num del tag Grups
Valors	Valors de les propietats	v index_node index_grup v1'v2'...	index_node: integer identificador del node index_grup: integer identificador del grup index_link: integer identificador del link v*: float valors de les propietats, separats per '.
		V index_link index_grup v1'v2'...	
Enllaços	Propietats dels enllaços	l index_link from_id to_id	l link unidireccional L link bidireccional index_link: integer identificador únic from_id: integer identificador del node inici el link to_id: integer identificador del node destinació del link
		L index_link from_id to_id	
Camins	Descripció de camins	p index num_nodes id1 id2 ...	p camí unidireccional P camí bidireccional index: integer identificador únic de camí num_nodes: integer
		P index num_nodes id1 id2 ...	

			número de nodes del camí id*: integer identificadors dels nodes que atravesa el camí, en ordre

Taula 8.7. Taula d'etiquetes del format d'entrada a Otter.

Un exemple de fitxer de configuració és el de la **Fig. 8.31**:

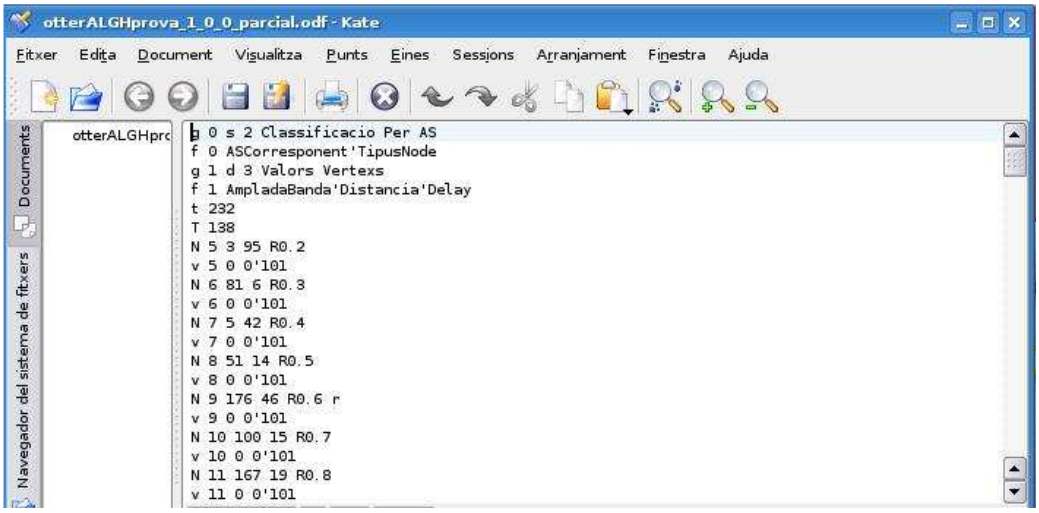


Fig. 8.31. Captura de l'arxiu d'entrada a Otter.

Per a més referències, consultar el manual de la pàgina d'Otter: http://www.caida.org/tools/visualization/otter/

8.12. Format de Fitxer de Resultats Estadístics de Simulacions

Com a sortida de l'aplicació desenvolupada s'obté un arxiu de text pla separat per tabulacions i que conté els aspectes més importants referents a una simulació i a la cerca d'un recurs a través de la xarxa. La **Fig. 8.32** mostra una captura de l'aspecte del fitxer generat.

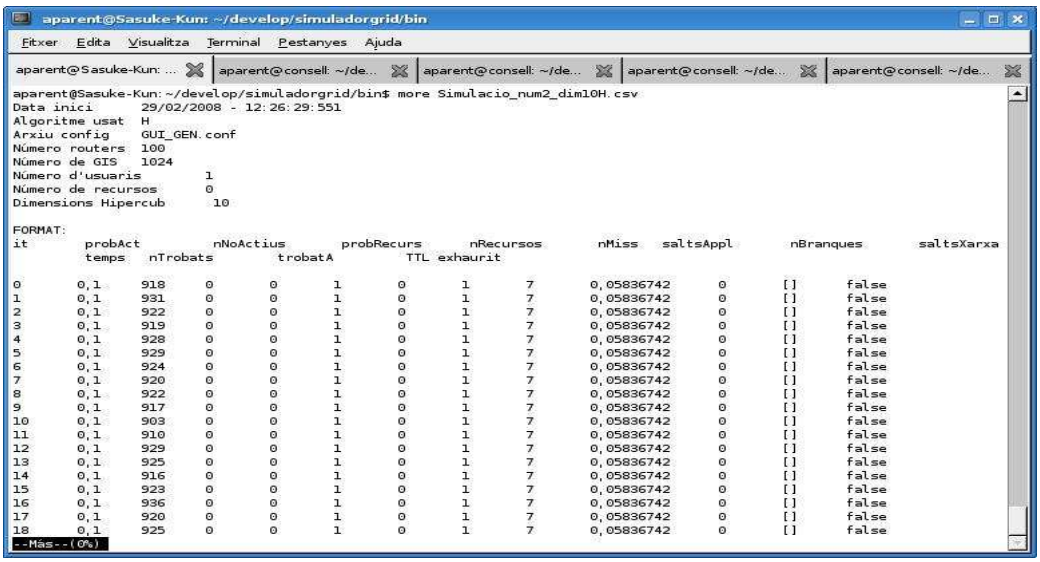


Fig. 8.32. Captura de l'arxiu de sortida de les simulacions.

Les dades que emmagatzema es poden dividir en dues parts: dades de configuració i informació general, i dades estadístiques. Les dades de configuració i informació emmagatzemades són les següents:

- **Data inici**. Data de l'inici de la simulació, la qual es mostra amb una precisió de milisegons.
- **Data finalització**. Ídem a la data d'inici, però refent a l'instant que ha acabat la simulació.
- **Algoritme usat**. Mostra quin algoritme ha estat usat en la simulació.
- **Arxiu de configuració**. Mostra el nom de l'arxiu de configuració de BRITE usat per a generar la topologia de xarxa de la simulació.
- **Número de routers**. Número de routers distribuïts sobre la xarxa.
- **Número de GIS**. Número de nodes Gis distribuïts al llarg de la xarxa.
- **Número d'usuaris**. Número d'usuaris distribuïts al llarg de la xarxa.
- **Número de recursos**. Número de recursos que es troben sobre la xarxa i dels quals en tenen coneixement els corresponents GIS.
- **Dimensions hipercub**. Dimensió de l'hipercub, que determina de forma directe el nombre de GIS que es troben distribuïts sobre la xarxa. A més, indica quants GIS estan interconnectats directament entre sí a nivell d'aplicació (veïns).

Per altra banda, la segona part de l'arxiu guarda les xifres estadístiques de cada una de les simulacions realitzades, això és:

- **Iteració**. Indica la iteració *i-èssima* realitzada amb els paràmetres de probabilitat d'activitat de GIS i probabilitat de posseir un recurs fixats. És a dir, per a cada combinació de les dues probabilitats esmentades, es realitzen *i* iteracions.

- **Probabilitat d'activitat del GIS**. Denota la probabilitat d'activitat de cada un dels GIS. S'aplica aquest factor just abans de realitzar la cerca d'un recurs. Els valors presos van des de 0,1 (probabilitat de tenir molts nodes inactius) fins a 1,0 (tots els nodes actius), amb increments de 0,1.

- **Número de GIS no actius al llarg de la xarxa**. A partir de la probabilitat d'activitat de GIS, es marquen els nodes no actius. Aquesta columna indica, per a cada simulació, quants GIS no estan actius.

- **Probabilitat que un GIS tingui un recurs**. Denota la probabilitat que un recurs sigui conegut per un GIS. Els valors presos van de de 0,1 (probabilitat baixa que un GIS tingui el recurs) fins a 1,0 (tots els GIS tenen el recurs), amb increments de 0,1.

- **Número de recursos distribuïts al llarg de la xarxa**. A partir de la probabilitat de recurs lligat a un GIS, mostra el número de recursos distribuïts sobre la xarxa.

- **Número de missatges processats per la xarxa**. Indica el número de missatges que s'han intercanviat dins la xarxa per a cada una de les simulacions.

- **Número de salts d'aplicació realitzats**. Mostra el número de salts entre GIS realitzats per a cada una dels simulacions.

- **Número de branques de cerca obertes**. Denota el nombre de camins de cerca que s'han obert a partir del missatge de petició inicial. La branca s'extèn fins que no hi ha més nodes als que reenviar el missatge de cerca o el node que l'ha rebut és un node inactiu.

- **Número de salts de xarxa (IP) realitzats**. Anàlogament a l'anterior ítem, mostra el número de salts a nivell de xarxa totals per a cada cerca de recurs. S'entèn com a salts de xarxa el salt entre dos routers contigus, entre un usuari i un router, i entre un GIS i un router.

- **Temps consumit en una simulació**. És el temps total que ha necessitat la cerca per acabar, sigui perquè ha trobat o no el recurs sol·licitat.

- **Número de recursos trobats**. En cas que durant la simulació s'hagi trobat el recurs sol·licitat, aquest ítem sera diferent de zero, indicant quantes vegades l'algoritme de cerca ha trobat el recurs. És un paràmetre relacionat amb les branques de cerca, ja que si en el primer node (node inicial) es troba el recurs sol·licitat, ja no es buscarà més i el procés de cerca finalitza retornant un recurs.

- **Localització del recurs trobat**. Mostra una llista dels GIS on s'hi ha trobat el recurs sol·licitat.

- **TTL exhaurit**. Indica mitjançant true/false si en aquesta iteració s'ha exhaurit o no el TTL. Aquest paràmetre impedeix, de la mateixa forma que a Internet, que un paquet entri en un bucle infinit.

8.13. Verificació de la Connectivitat Extrem a Extrem

Abans de passar a la implementació dels algoritmes de cerca cal assegurar-se que la connectivitat entre tots els elements de la xarxa és satisfactòria, és a dir, que entre tots els nodes sigui possible l'intercanvi de missatges. A més, s'ha de comprovar que aquest intercanvi es realitzi usant els camins de xarxa més curts entre dos punts, de manera que s'assegura que l'algoritme de routing ha convergit correctament.

La comprovació de la connectivitat es realitza mitjançant successives proves usant l'eïna ping, que envia un missatge des d'un node inicial (usuari) fins al node destí (per exemple un GIS), guardant el pas pels routers intermitjos formant el camí d'anada i tornada. Si aquest paquet és capaç de retornar a l'origen, indica que la topologia està correctament interconnectada i que l'algoritme de routing és capaç de determinar el camí des de l'origen fins al destí.

Un exemple de connectivitat és el següent, en el que es mostra un ping enviat des de U0.0 fins a G9.9 (**Fig. 8.33**):

-------- PING DES DE U0.0 A G9.9 ----------
Ping information for U0.0

Entity Name	Entry Time	Exit Time	Bandwidth----
U0.0	800,000	800,0001	10000000,000
R0.10	800,0234	800,0234	10000000000000,000
R0.25	800,0243	800,0243	10000000000000,000
R0.19	800,0245	800,0245	100000000000000,000
R8.181	800,0248	800,0248	10000000000000,000
R8.172	800,0259	800,0259	10000000000000,000
R8.170	800,0261	800,0261	100000000000000,000
R9.201	800,0263	800,0263	10000000000000,000
R9.209	800,027	800,027	10000000000000,000
R9.190	800,0281	800,0281	100000000,000
G9.9	800,055	800,055	100000000,000
R9.190	800,0819	800,0819	10000000000000,000
R9.209	800,0829	800,0829	10000000000000,000
R9.201	800,0836	800,0836	100000000000000,000
R8.170	800,0838	800,0838	10000000000000,000
R8.172	800,084	800,084	10000000000000,000
R8.181	800,0851	800,0851	100000000000000,000
R0.19	800,0854	800,0854	10000000000000,000
R0.25	800,0856	800,0856	10000000000000,000
R0.10	800,0865	800,0866	10000000,000
U0.0	800,1099	N/A	N/A

Round Trip Time : 0,1098 seconds
Number of Hops : 10
Bottleneck Bandwidth : 1.0E7 bits/s
-------- FI DE PING ----------

Fig. 8.33. Resultat del ping des de U0.0 a G9.9.

Es pot observar el camí seguit des de U0.0 (usuari 0, que està situat a l'AS 0) fins arribar al G9.9, i com el missatge de ping recorre els routers de l'AS 0 fins a l'AS 9 (AS destí), passant per l'AS 8, que en aquest cas fa d'intermediari entre les dues subxarxes (veure 4.5.3.). Així doncs, es verifica que la connectivitat és correcte i que a més el routing funciona satisfactoriament tant dins de cada AS com entre ells mateixos.

8.14. Verificació de la Implementació dels Algoritmes de Cerca

En aquest annex es troben les imatges que demostren el funcionament paral·lel dels algoritmes implementats respecte de la demostració teòrica corresponent. Per a cada algoritme, es mostra una imatge per a cada branca de cerca dins del GRID, on es pot comprovar el recorregut que segueix el missatge a nivell d'aplicació (i a nivell IP) al llarg de la xarxa.

8.14.1. Algoritme P

BrancaID=1

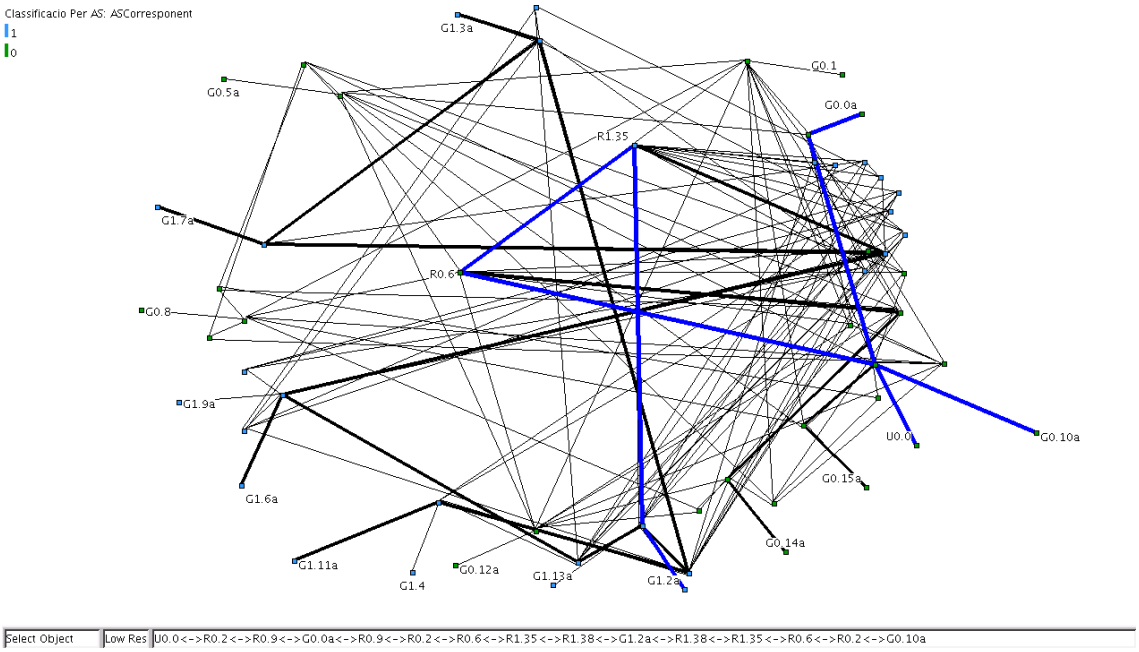


Fig. 8.34. Recorregut BrancaID 1 amb algoritme P.

BrancaID=2

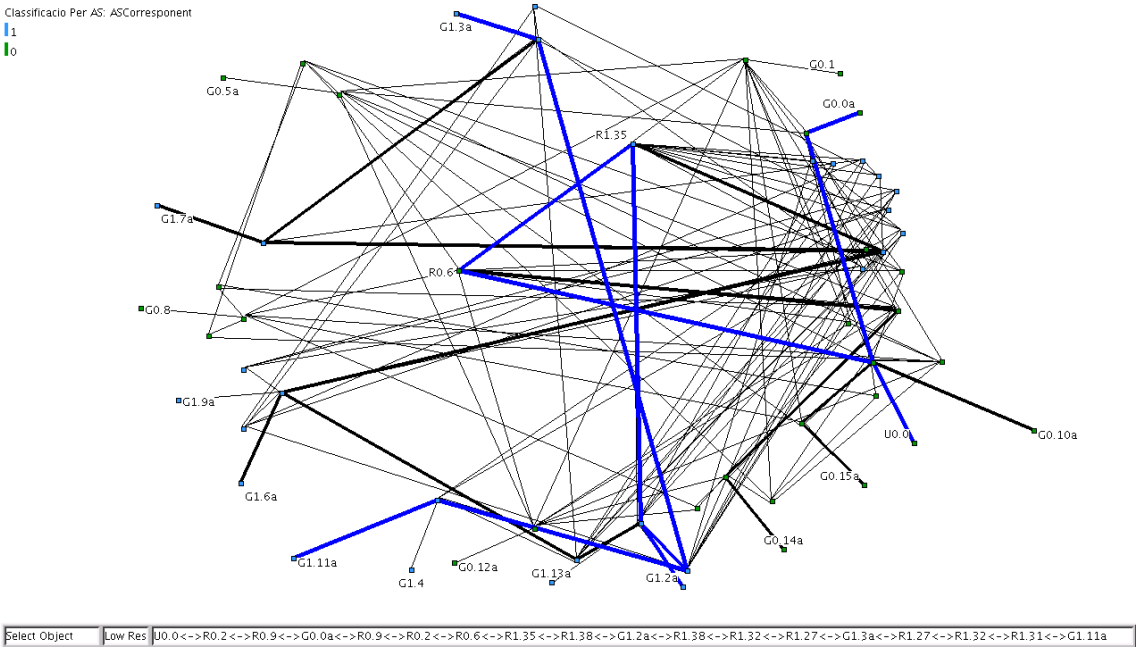


Fig. 8.35. Recorregut BrancaID 2 amb algoritme P.

BrancaID=3

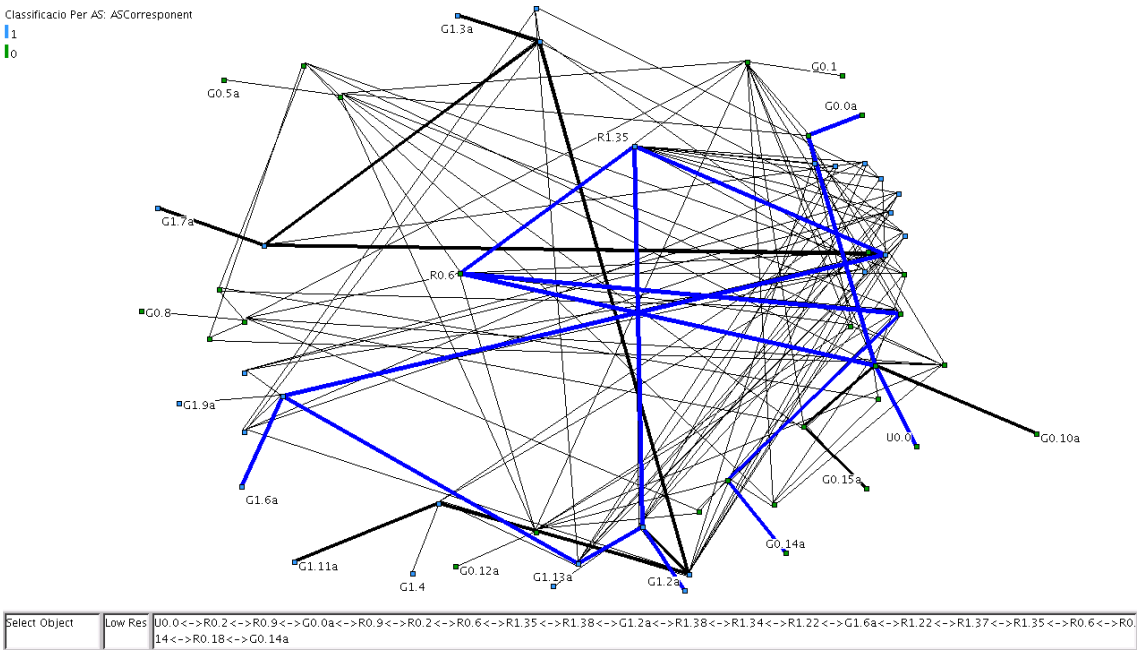


Fig. 8.36. Recorregut BrancaID 3 amb algoritme P.

BrancaID=4

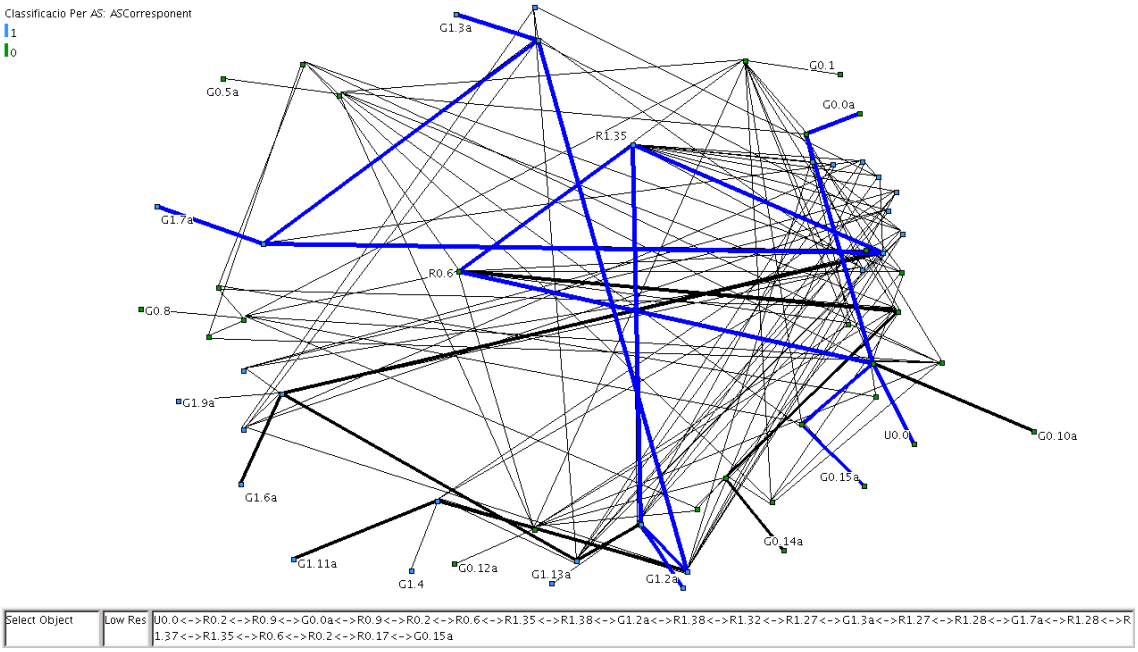


Fig. 8.37. Recorregut BrancaID 4 amb algoritme P.

8.14.2. Algoritme H

BrancaID=1

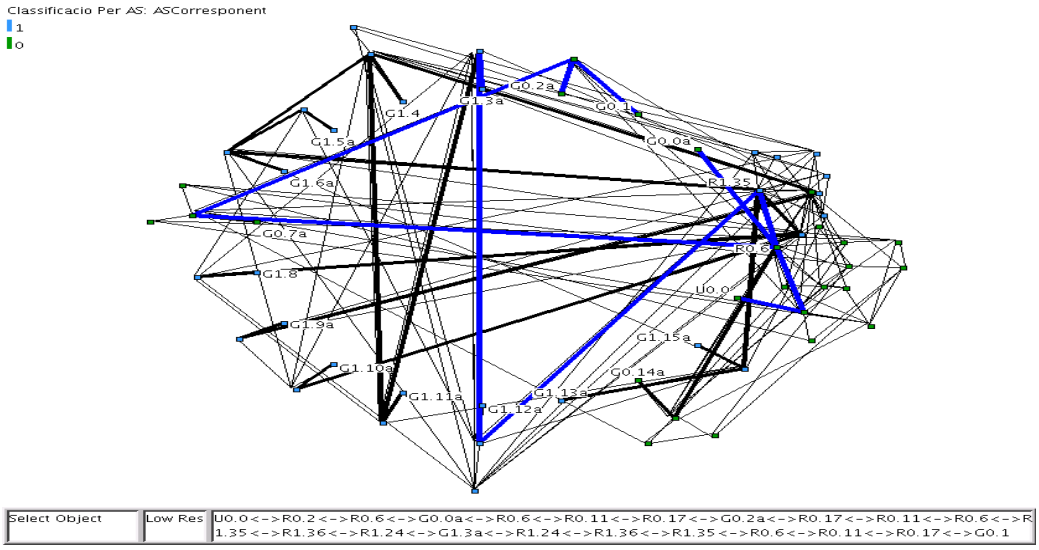


Fig. 8.38. Recorregut de BrancaID 1 amb algoritme H.

BrancaID=2

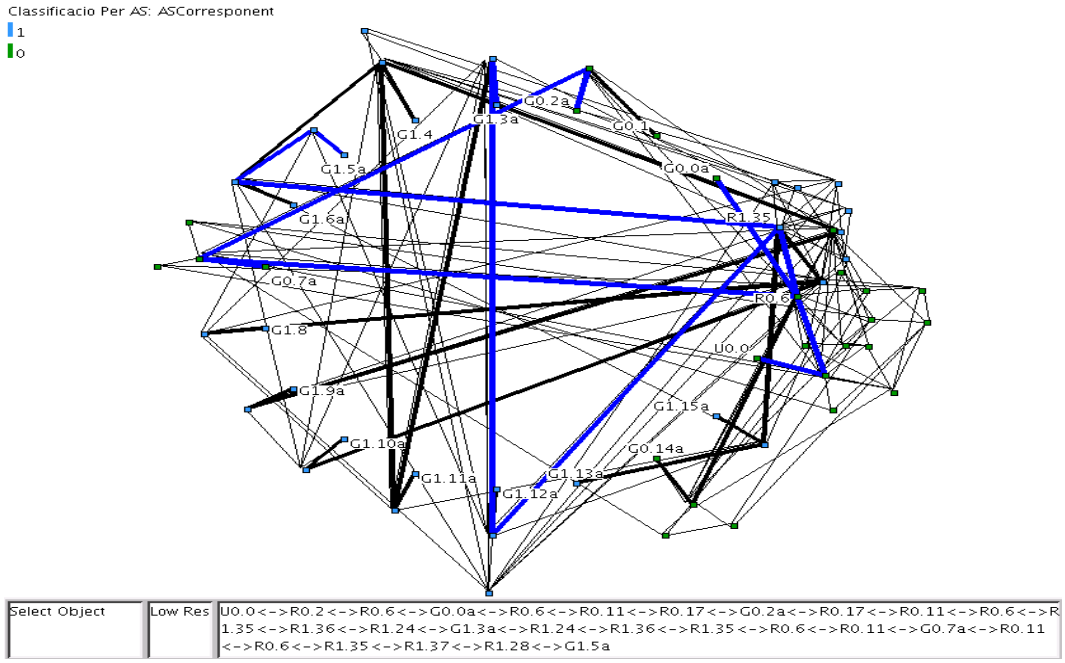


Fig. 8.39. Recorregut de BrancaID 2 amb algoritme H.

BrancaID=3

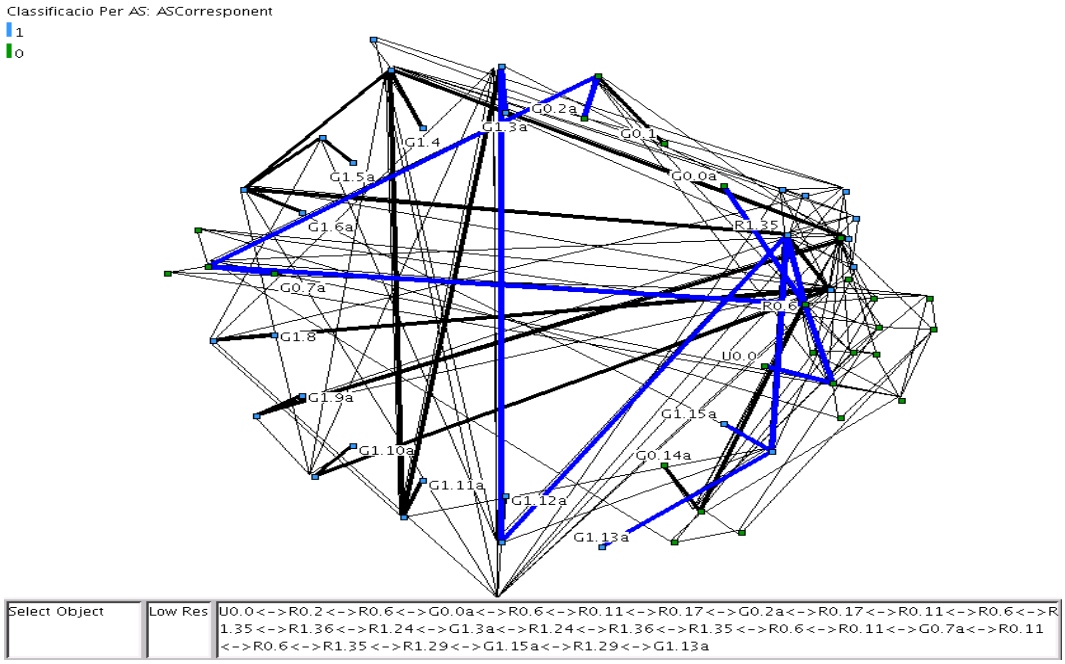


Fig. 8.40. Recorregut de BrancaID 3 amb algoritme H.

BrancaID=4

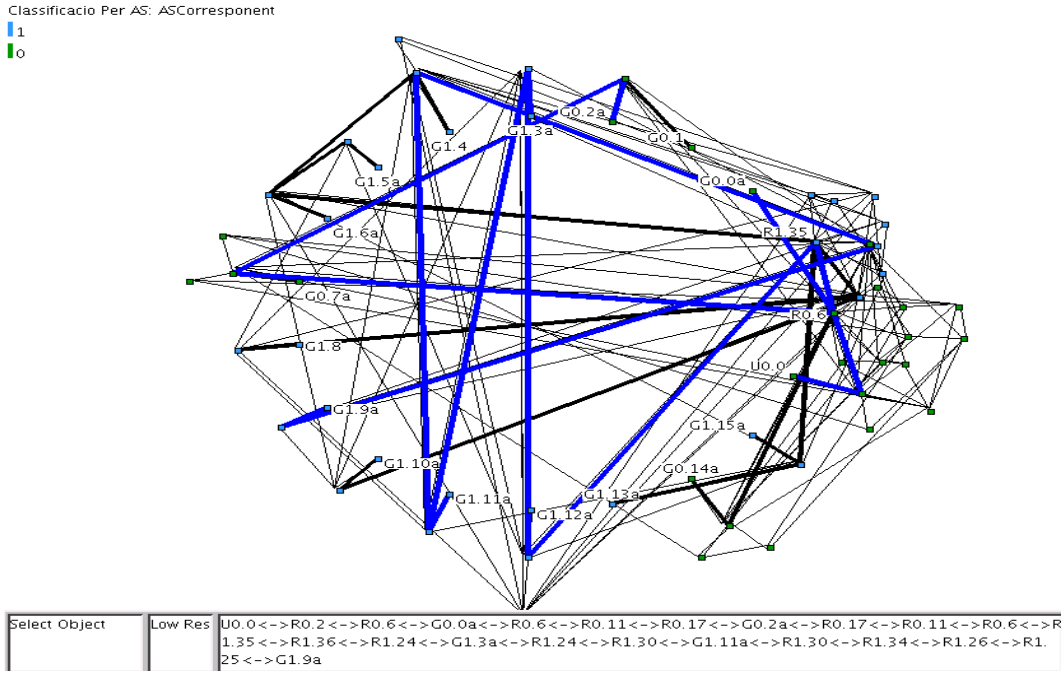


Fig. 8.41. Recorregut de BrancaID 4 amb algoritme H.

BrancaID=5

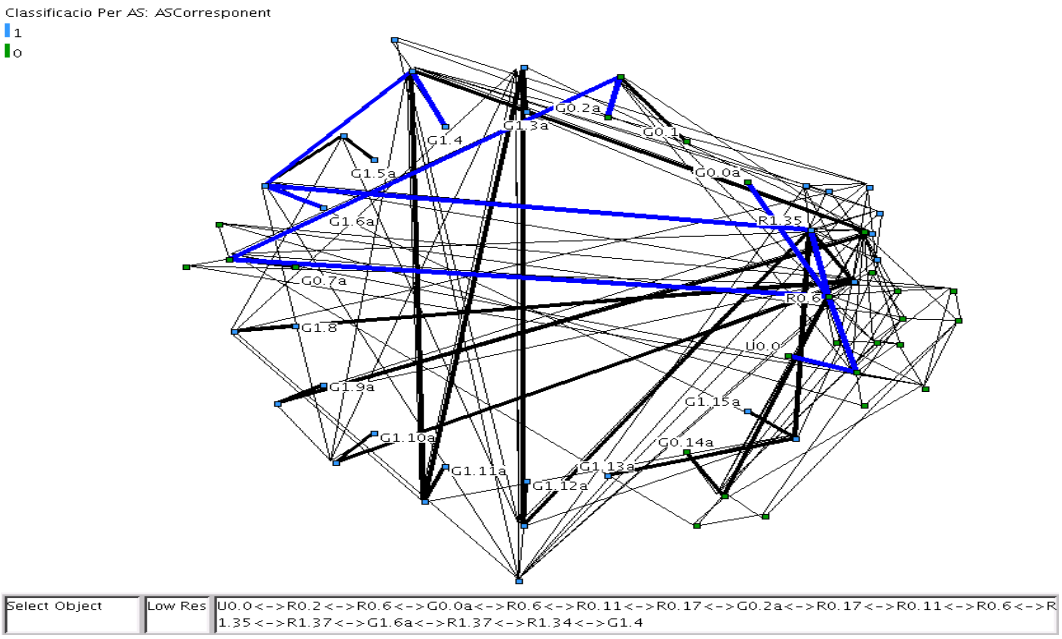


Fig. 8.42. Recorregut de BrancaID 5 amb algoritme H.

BrancaID=6

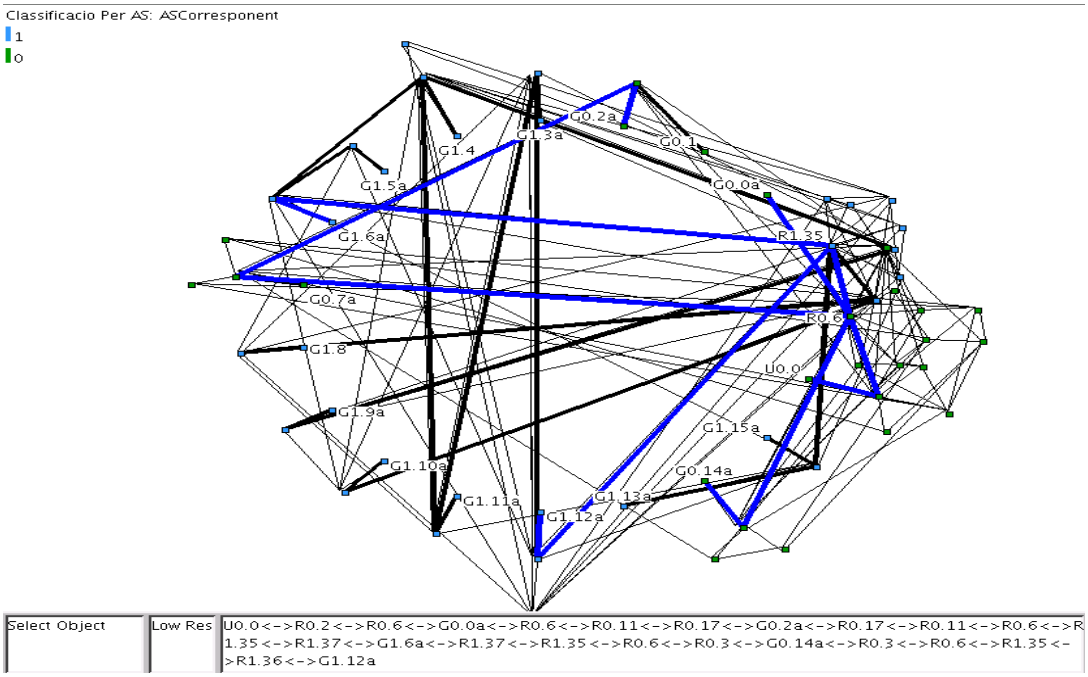


Fig. 8.43. Recorregut de BrancaID 6 amb algoritme H.

BrancaID=7

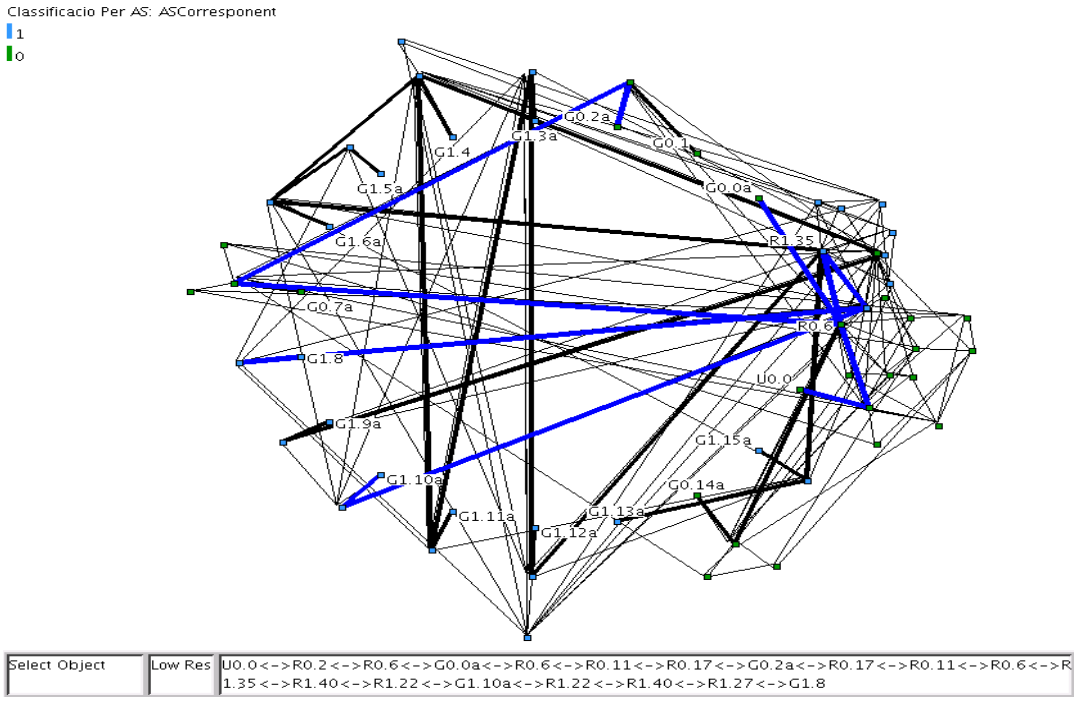


Fig. 8.44. Recorregut de BrancaID 7 amb algoritme H.

8.15. Taules de Dades i Gràfiques

8.15.1. Primer Grup de Simulacions

		1		2		3		4		5		6		7		8		9		10	
		P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H
Missatges d'aplicació	Mínim	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Màxim	2	2	4	4	8	8	16	16	32	32	64	64	128	128	256	256	512	512	1024	1024
	Mitjana	1,19	1,21	1,52	1,52	1,89	1,98	2,92	2,93	4,56	4,75	7,57	8,11	12,61	13,76	22,90	24,26	39,49	45,18	75,33	84,73
	Desv típica	0,40	0,41	1,01	0,99	1,89	2,00	3,89	3,95	7,48	7,82	14,05	15,04	26,93	28,66	51,61	54,94	97,60	106,28	190,42	204,84
Salts Branques	Mínim	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Màxim	1	1	2	2	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11
	Mitjana	0,19	0,21	0,37	0,37	0,44	0,48	0,69	0,69	0,88	0,90	1,12	1,15	1,25	1,34	1,50	1,52	1,67	1,79	1,98	2,09
	Desv típica	0,40	0,41	0,70	0,69	0,89	0,94	1,27	1,31	1,63	1,68	1,99	2,08	2,31	2,44	2,72	2,79	3,05	3,20	3,44	3,64
Branques	Mínim	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Màxim	1	1	2	2	4	4	8	8	16	16	32	32	64	64	128	128	256	256	512	512
	Mitjana	1	1	1,16	1,15	1,40	1,43	1,84	1,85	2,57	2,72	3,98	4,33	6,36	7,03	11,04	12,01	18,54	21,96	34,34	40,81
	Desv típica	0	0	0,36	0,36	0,88	0,91	1,80	1,80	3,48	3,62	6,64	7,05	12,71	13,31	24,18	25,51	45,59	49,41	87,65	94,83
Salts Xarxa	Mínim	8	7	7	8	8	8	8	7	6	4	8	8	7	6	8	8	3	7	8	7
	Màxim	18	14	39	46	107	110	225	220	431	446	1214	1237	2443	2445	5783	6006	10442	11854	26617	24438
	Mitjana	9,94	8,46	12,29	14,38	20,66	20,39	33,34	30,00	47,70	49,77	116,15	122,67	198,01	226,05	429,28	502,34	672,81	934,56	1637,51	1860,28
	Desv típica	3,95	2,84	10,53	12,47	27,29	26,08	53,90	50,58	94,48	99,84	249,01	251,05	476,10	511,29	1065,50	1197,55	1833,74	2285,30	4485,07	4583,78
Temps	Mínim	0,08	0,06	0,08	0,09	0,09	0,05	0,07	0,07	0,11	0,06	0,08	0,11	0,05	0,05	0,05	0,09	0,08	0,07	0,06	0,06
	Màxim	0,18	0,11	0,30	0,27	0,34	0,31	0,40	0,40	0,53	0,45	0,52	0,55	0,47	0,44	0,44	0,56	0,71	0,76	0,47	0,55
	Mitjana	0,10	0,07	0,12	0,12	0,13	0,07	0,11	0,11	0,16	0,09	0,14	0,17	0,10	0,09	0,11	0,16	0,14	0,14	0,12	0,12
	Desv típica	0,04	0,02	0,07	0,05	0,07	0,05	0,08	0,07	0,10	0,07	0,10	0,10	0,09	0,07	0,09	0,12	0,11	0,11	0,10	0,10
Recursos Trobats	Mínim	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Màxim	1	1	2	2	4	4	7	7	11	12	17	21	28	29	58	60	92	81	153	155
	Mitjana	0,34	0,34	0,41	0,44	0,52	0,53	0,74	0,72	1,02	1,03	1,54	1,55	2,31	2,31	3,75	3,49	5,66	5,73	9,55	10,03
	Desv típica	0,47	0,47	0,55	0,56	0,72	0,72	1,12	1,07	1,68	1,75	2,81	2,93	4,68	4,64	8,14	7,59	13,26	13,05	22,33	23,18

Taula 8.8. Simulació amb 100 routers distribuïts entre 10 AS

		1		2		3		4		5		6		7		8		9		10	
		P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H
Missatges d'aplicació	Mínim	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Màxim	2	2	4	4	8	8	16	16	32	32	64	64	128	128	256	256	512	512	1024	1024
	Mitjana	1,18	1,19	1,50	1,49	2,06	1,97	2,92	2,94	4,43	4,81	7,43	8,02	12,53	14,18	23,32	24,89	40,57	45,06	78,37	83,77
	Desv típica	0,39	0,39	0,98	0,97	2,06	1,96	3,94	3,97	7,36	7,94	14,11	15,01	26,68	28,76	51,82	55,09	98,74	106,75	193,27	209,07
Salts Branques	Mínim	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Màxim	1	1	2	2	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11
	Mitjana	0,18	0,19	0,35	0,35	0,53	0,47	0,70	0,70	0,85	0,92	1,06	1,14	1,26	1,40	1,56	1,57	1,70	1,75	2,03	2,02
	Desv típica	0,39	0,39	0,69	0,69	0,99	0,91	1,32	1,32	1,61	1,72	1,96	2,09	2,33	2,48	2,74	2,81	3,08	3,16	3,50	3,56
Branques	Mínim	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Màxim	1	1	2	2	4	4	8	8	16	16	32	32	64	64	128	128	256	256	512	512
	Mitjana	1	1	1,15	1,14	1,45	1,43	1,90	1,85	2,69	2,73	4,26	4,27	6,85	7,23	12,37	12,48	21,05	22,00	40,11	40,35
	Desv típica	0	0	0,36	0,35	0,93	0,92	1,89	1,81	3,63	3,67	7,07	7,01	13,35	13,45	25,89	25,76	49,05	49,85	95,81	96,90
Salts Xarxa	Mínim	10	10	10	9	10	7	10	9	9	7	10	7	10	10	9	9	9	10	9	10
	Màxim	19	20	47	38	104	100	238	284	521	533	1145	1090	2807	2683	5675	5580	12542	13602	26449	28354
	Mitjana	11,65	11,85	15,85	13,71	24,03	17,83	38,21	37,14	62,34	57,98	121,09	111,34	244,66	264,40	474,21	492,01	932,60	1077,73	1912,13	2113,07
	Desv típica	3,49	3,89	12,00	9,65	28,06	22,83	59,16	59,87	118,25	112,33	249,38	234,00	564,32	568,99	1114,27	1133,00	2365,24	2639,76	4865,43	5375,43
Temps	Mínim	0,09	0,10	0,10	0,07	0,06	0,08	0,10	0,06	0,10	0,12	0,10	0,10	0,04	0,09	0,05	0,10	0,10	0,05	0,12	0,10
	Màxim	0,19	0,17	0,29	0,22	0,28	0,42	0,47	0,47	0,50	0,56	0,58	0,53	0,42	0,53	0,45	0,59	0,63	0,54	0,63	0,73
	Mitjana	0,11	0,11	0,13	0,09	0,08	0,12	0,15	0,11	0,15	0,18	0,15	0,17	0,07	0,16	0,10	0,18	0,16	0,11	0,19	0,20
	Desv típica	0,04	0,03	0,07	0,03	0,04	0,07	0,09	0,09	0,10	0,11	0,10	0,12	0,06	0,11	0,08	0,14	0,11	0,10	0,12	0,16
Recursos Trobats	Mínim	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Màxim	1	1	2	2	4	4	7	7	11	13	20	18	32	30	53	55	85	83	146	148
	Mitjana	0,33	0,33	0,42	0,40	0,56	0,52	0,73	0,71	1,02	1,00	1,52	1,51	2,19	2,35	3,79	3,91	5,67	5,85	10,43	9,62
	Desv típica	0,47	0,47	0,55	0,54	0,76	0,73	1,08	1,10	1,72	1,74	2,88	2,79	4,44	4,69	8,13	8,32	12,92	13,47	24,04	22,77

Taula 8.9. Simulació amb 200 routers distribuïts entre 10 AS.

RELACIONS ENTRE MÈTRIQUES MITJANES																				
Dimensió	1		2		3		4		5		6		7		8		9		10	
Algoritme	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H
MissAplica/ Recursos_10	3,51	3,59	3,70	3,47	3,67	3,75	3,97	4,10	4,48	4,61	4,93	5,23	5,47	5,97	6,11	6,94	6,97	7,88	7,89	8,45
Recursos/ Branques_10	0,34	0,34	0,36	0,38	0,37	0,37	0,40	0,39	0,40	0,38	0,39	0,36	0,36	0,33	0,34	0,29	0,31	0,26	0,28	0,25
SaltsXarxa/ Branques_10	9,94	8,46	10,62	12,51	14,75	14,29	18,15	16,24	18,53	18,29	29,20	28,35	31,14	32,17	38,90	41,82	36,52	42,55	47,68	45,58
SaltsXarxa/ MissAplica_10	8,32	7,00	8,07	9,47	10,93	10,31	11,42	10,24	10,47	10,47	15,34	15,12	15,70	16,43	18,75	20,71	17,14	20,69	21,74	21,95
SaltsXarxa/ Recursos_10	29,22	25,11	29,90	32,88	40,12	38,67	45,36	41,93	46,91	48,29	75,57	79,03	85,87	98,07	114,50	143,75	119,54	162,98	171,52	185,55
MissAplica/ Branques_10	1,19	1,21	1,32	1,32	1,35	1,39	1,59	1,59	1,77	1,75	1,90	1,88	1,98	1,96	2,08	2,02	2,13	2,06	2,19	2,08

Taula 8.10. Relacions entre valors mitjans de les mètriques. 100 routers distribuïts en 10AS.

RELACIONS ENTRE MÈTRIQUES MITJANES																				
Dimensió	1		2		3		4		5		6		7		8		9		10	
Algoritme	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H
MissAplica/ Recursos_20	3,57	3,64	3,59	3,69	3,65	3,80	4,02	4,14	4,33	4,81	4,89	5,32	5,71	6,02	6,16	6,37	7,16	7,70	7,52	8,70
Recursos/ Branques_20	0,33	0,33	0,36	0,35	0,39	0,36	0,38	0,38	0,38	0,37	0,36	0,35	0,32	0,33	0,31	0,31	0,27	0,27	0,26	0,24
SaltsXarxa/ Branques_20	11,65	11,85	13,80	12,01	16,62	12,43	20,10	20,03	23,16	21,25	28,41	26,05	35,70	36,59	38,33	39,42	44,30	48,99	47,68	52,37
SaltsXarxa/ MissAplica_20	9,84	10,00	10,58	9,19	11,69	9,07	13,07	12,62	14,07	12,06	16,29	13,89	19,53	18,65	20,34	19,77	22,99	23,92	24,40	25,22
SaltsXarxa/ Recursos_20	35,12	36,37	38,00	33,89	42,64	34,45	52,57	52,24	60,95	58,09	79,66	73,87	111,58	112,29	125,21	125,94	164,52	184,10	183,36	219,56
MissAplica/ Branques_20	1,18	1,19	1,30	1,31	1,42	1,37	1,54	1,59	1,65	1,76	1,74	1,88	1,83	1,96	1,88	1,99	1,93	2,05	1,95	2,08

Taula 8.11. Relacions entre valors mitjans de les mètriques. 200 routers distribuïts en 20AS.

8.15.2. Segon Grup de Simulacions

		Sim1		Sim2		Sim3		Sim4		Sim5		Sim6		Sim7		Sim8		Sim9	
		8		8		8		8		8		8		8		8		8	
		P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H
Missatges d'aplicació	Mínim	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Màxim	256	256	256	256	256	256	256	256	256	256	256	256	256	256	256	256	256	256
	Mitjana	23,02	24,56	24,00	25,35	23,62	24,02	23,02	24,19	21,98	26,00	21,69	24,10	23,49	25,86	22,15	23,86	22,14	24,12
	Desv típica	51,67	54,52	53,16	55,23	52,42	54,27	51,44	54,43	51,16	55,57	50,68	54,45	52,23	56,18	51,29	53,60	51,09	54,73
Salts Branques	Mínim	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Màxim	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9	8	9
	Mitjana	1,53	1,58	1,59	1,63	1,54	1,55	1,54	1,53	1,44	1,66	1,46	1,52	1,57	1,65	1,49	1,53	1,51	1,51
	Desv típica	2,73	2,82	2,77	2,87	2,74	2,83	2,72	2,81	2,66	2,89	2,68	2,79	2,76	2,87	2,68	2,80	2,68	2,78
Branques	Mínim	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Màxim	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128
	Mitjana	12,22	12,18	12,61	12,53	12,51	11,90	12,24	12,10	11,68	12,83	11,52	11,95	12,43	12,82	11,80	12,02	11,82	11,97
	Desv típica	25,77	25,34	26,31	25,60	26,14	25,43	25,62	25,52	25,50	25,80	25,21	25,31	25,99	26,07	25,57	25,15	25,52	25,48
Salts Xarxa	Mínim	8	8	10	8	10	7	6	9	10	7	7	4	7	7	4	4	7	4
	Màxim	5873	5905	5931	5427	5583	5248	4585	5596	5172	5172	4278	3487	3969	3816	3495	3456	4000	3735
	Mitjana	483,10	494,32	508,83	479,41	481,78	439,67	375,65	465,44	410,30	457,05	330,09	299,63	338,93	350,74	271,34	288,62	319,72	307,79
	Desv típica	1151,62	1155,79	1186,14	1089,97	1118,79	1050,50	893,14	1100,82	1001,13	1029,97	816,78	707,24	790,54	789,69	674,03	681,31	777,76	739,17
Temps	Mínim	0,06	0,08	0,11	0,10	0,12	0,08	0,05	0,08	0,08	0,11	0,09	0,10	0,08	0,07	0,09	0,09	0,11	0,07
	Màxim	0,63	0,57	0,66	0,55	0,79	0,59	0,50	0,52	0,52	0,68	0,56	0,56	0,49	0,61	0,55	0,66	0,58	0,57
	Mitjana	0,13	0,14	0,19	0,18	0,20	0,16	0,11	0,15	0,15	0,18	0,15	0,18	0,14	0,11	0,15	0,16	0,18	0,13
	Desv típica	0,11	0,10	0,13	0,12	0,13	0,14	0,10	0,13	0,12	0,13	0,10	0,14	0,09	0,08	0,11	0,13	0,12	0,11
Recursos Trobats	Mínim	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Màxim	49	51	59	50	53	52	53	52	48	61	58	53	47	50	53	59	50	53
	Mitjana	3,73	3,69	3,62	3,80	3,72	3,44	3,75	3,56	3,48	3,80	3,41	3,49	3,65	3,84	3,61	3,74	3,58	3,52
	Desv típica	7,89	7,85	7,63	7,99	7,91	7,45	8,03	7,81	7,77	7,89	7,49	7,54	7,79	7,88	7,85	8,02	7,63	7,63

Taula 8.12. Hipercub r= 8, variant la disposició de la topologia IP.

RELACIONS ENTRE MÈTRIQUES MITJANES																		
Simulació	Sim1		Sim2		Sim3		Sim4		Sim5		Sim6		Sim7		Sim8		Sim9	
Dimensió	8		8		8		8		8		8		8		8		8	
Algoritme	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H
MissAplica/ Recursos	6,17	6,65	6,64	6,66	6,35	6,99	6,14	6,79	6,32	6,84	6,35	6,91	6,44	6,74	6,13	6,38	6,19	6,84
Recursos/ Branques	0,31	0,30	0,29	0,30	0,30	0,29	0,31	0,29	0,30	0,30	0,30	0,29	0,29	0,30	0,31	0,31	0,30	0,29
SaltsXarxa/ Branques	39,53	40,60	40,36	38,25	38,51	36,94	30,69	38,48	35,12	35,61	28,66	25,08	27,26	27,37	22,99	24,01	27,05	25,71
SaltsXarxa/ MissAplica	20,99	20,12	21,20	18,91	20,40	18,31	16,32	19,24	18,66	17,58	15,22	12,43	14,43	13,56	12,25	12,10	14,44	12,76
SaltsXarxa/ Recursos	129,60	133,81	140,72	126,01	129,56	127,91	100,15	130,69	117,99	120,15	96,70	85,89	92,89	91,45	75,14	77,14	89,42	87,34
MissAplica/ Branques	1,88	2,02	1,90	2,02	1,89	2,02	1,88	2,00	1,88	2,03	1,88	2,02	1,89	2,02	1,88	1,98	1,87	2,01

Taula 8.13. Relacions entre valors mitjans de les mètriques. Hipercub de r=8, variant la topologia IP.

8.16. Anàlisi de Dades Estadístiques

En aquest annexe es realitza un anàlisi de les dades contingudes en les taules **Taula 8.8** a **Taula 8.13** mitjançant l'ús de múltiples gràfiques. Aquests anàlisis condueixen a les conclusions establertes en el cos del document.

Inicialment, es realitza una comparació d'algunes relacions importants entre mètriques en funció de la dimensió de l'hipercub i l'algoritme de cerca, prenint els corresponents valors mitjans. Les relacions comparades són les següents:

- **Salts de xarxa / Branques de cerca**. Indica quants de salts de xarxa acumula una branca en mitjana.

- **Missatges d'aplicació / Recursos trobats**. Denota el número de missatges d'aplicació, en mitjana, necessaris per a trobar un recurs dins del GRID.

- **Recursos trobats / Branques**. Mostra el número de recursos trobats per a cada branca de cerca oberta dins del GRID.

- **Missatges d'aplicació / Branques**. Indica el número de missatges a nivell d'aplicació que acumula en mitjana una brancad de cerca.

- **Salts de xarxa / Missatges d'aplicació**. Mesura la relació entre el número de salts de xarxa que acumula cada missatges a nivell d'aplicació.

- **Salts de xarxa / Recursos trobats**. Indica el número de salts a nivell IP necessaris en mitjana per a trobar un recurs.

En les gràfiques de **Fig. 8.45** i **Fig. 8.46**, totes les mètriques es pinten en l'eix d'ordenades secundari, excepte "SaltsXarxa/Recursos" i "SaltsXarxa/Branques. En l'eix X s'indica per a cada grup de valos graficats, a quina dimensió i algoritme es correspon. Per exemple, 4H indica que l'hipercub és de dimensió 4 i el procés de cerca es realitza mitjançant l'algoritme H. Finalment, el sufix "_10" indica que la simulació es correspon a la composta per 10AS a nivell IP.

Si s'observen les dues gràfiques, a simple vista es veure com cada una de les mètriques segueix una tendència similar en ambdós gràfics. Aquest fet indica ja que, per valors mitjans, la variació de la topologia de xarxa no afecta al rendiment dels algoritmes de cerca a nivell d'aplicació. L'única relació entre mètriques que varia de forma més important és la dels salts de xarxa respecte dels recursos trobats a mesura que la dimensió de l'hipercub va augmentant. La variació ve donada per l'augment en el nombre de salts de xarxa acumulats en el cas de la topologia amb 200 routers, mentre que el nombre de recursos trobats segueix sent el mateix que en la topologia de 100 routers. En canvi, la resta de mètriques presenten aproximadament els mateixos valors.

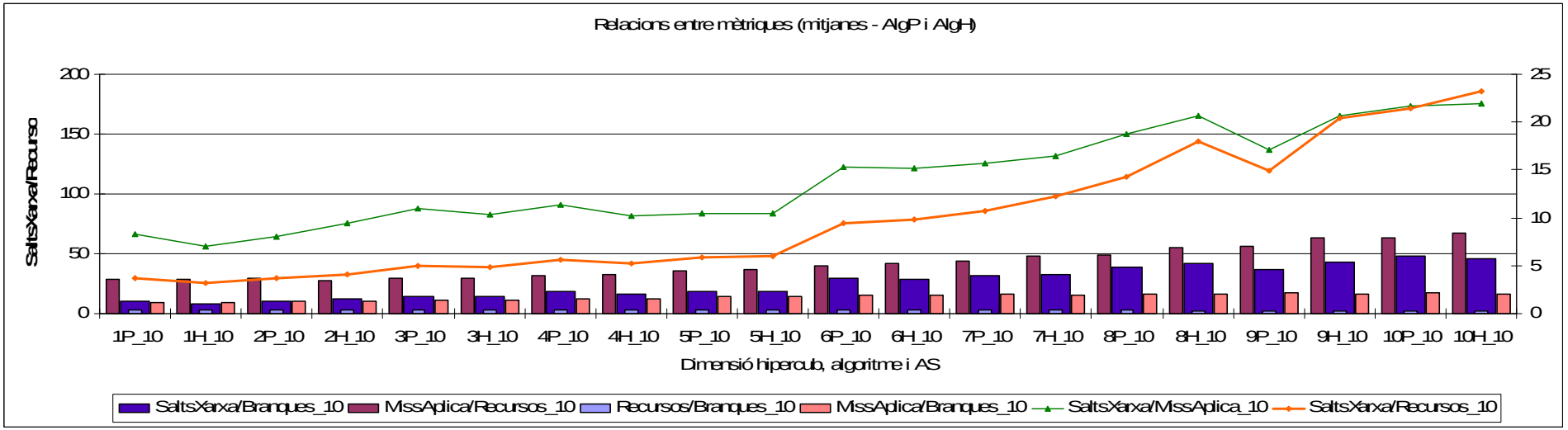


Fig. 8.45. Relacions entre mètriques mitjanes, en funció de la dimensió de l'hipercub i l'algoritme. Composoció de 10AS.

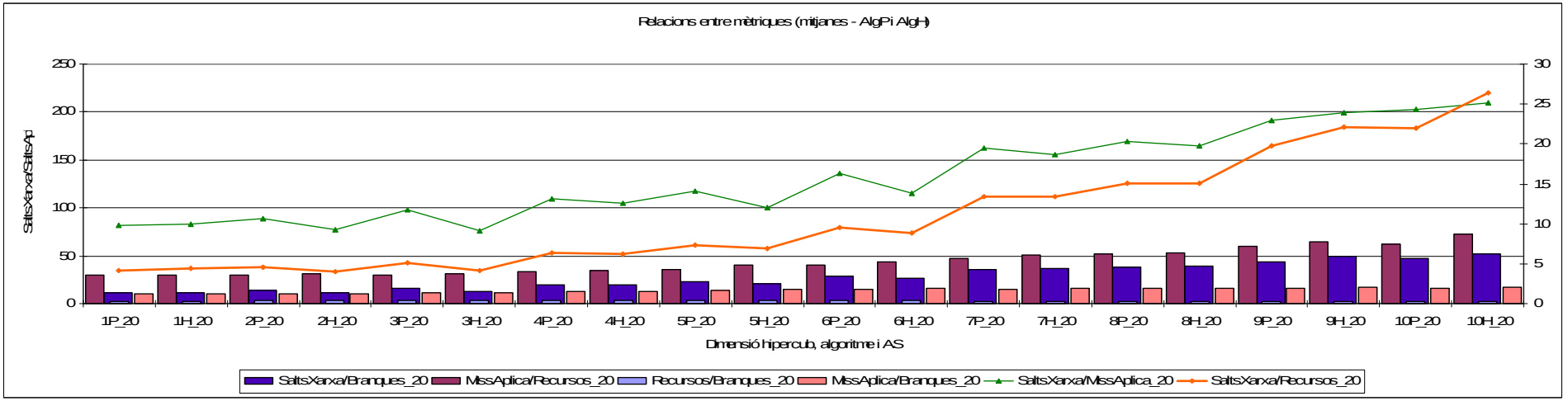


Fig. 8.46. Relacions entre mètriques mitjanes, en funció de la dimensió de l'hipercub i l'algoritme. Composoció de 20AS.

Les següents gràfiques (de **Fig. 8.47** a **Fig. 8.51**) expresen la relació de cada una de les mètriques analitzades respecte de la probabilitat d'activitat de GIS i de la probabilitat de conèixer recurs, havent fixat la dimensió de l'hipercub a 10. Cada gràfica representa un procés complet de cerca, escombrant totes les combinacions possibles de parelles de probabilitats. Les gràfiques del costat esquerre corresponen a la topologia IP amb 100 routers, mentre que les de la dreta amb topologia IP de 200 routers.

Les àrees en segon pla escalonades es corresponen amb les probabilitats de conèixer recurs, mentre que la línia també escalonada representa la probabilitat d'activitat dels GIS. Els valors es representen en format de gràfic de dispersió, on els punts vermells representen dades referents a l'algoritme H, mentre que les blaves són les corresponents a l'algoritme P.

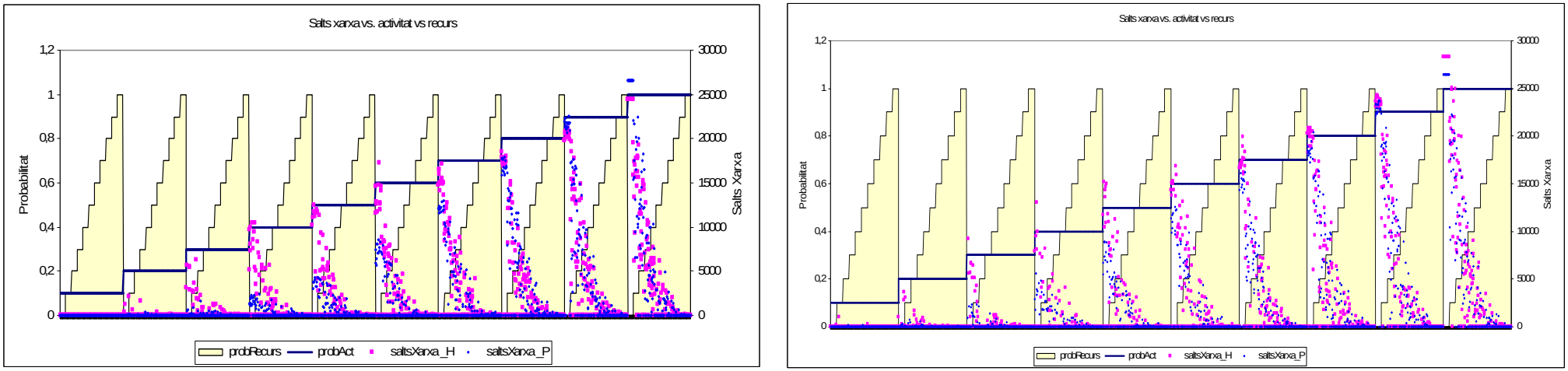


Fig. 8.47. Salts de xarxa per hipercub de dimensió 10. 100 i 200 routers.

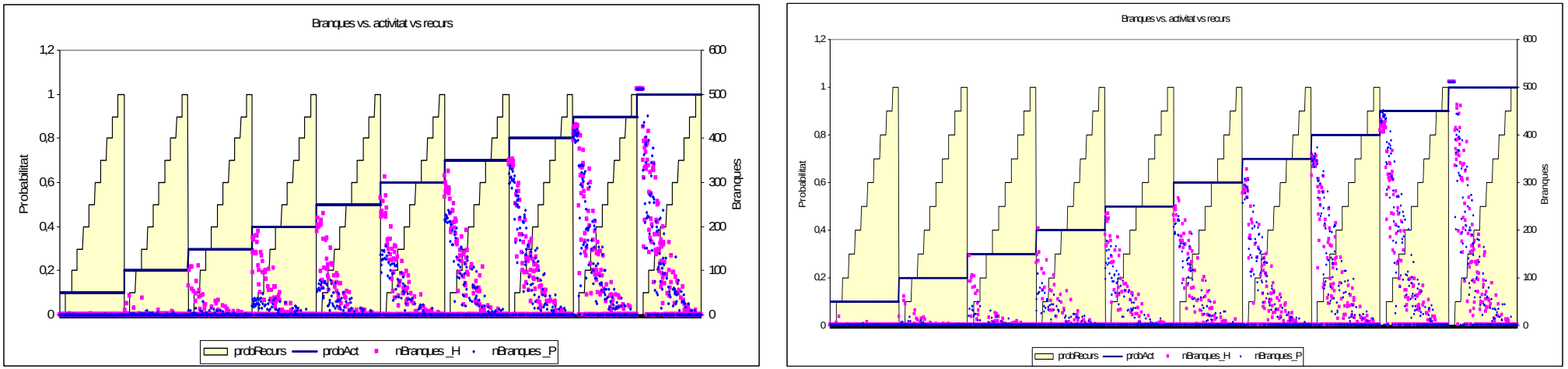


Fig. 8.48. Branques de cerca per hipercub de dimensió 10. 100 i 200 routers.

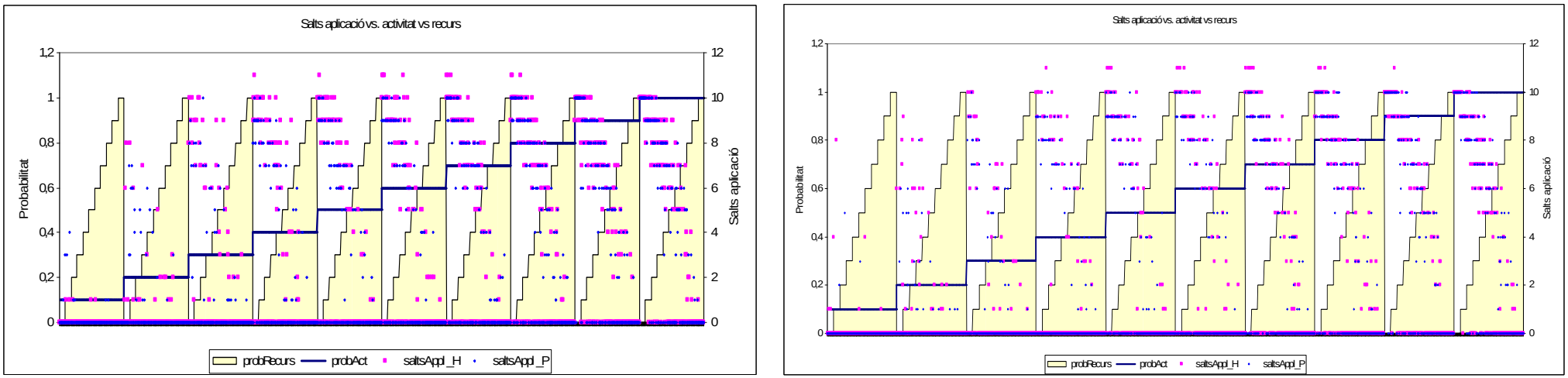


Fig. 8.49. Salts d'aplicació per hipercub de dimensió 10. 100 i 200 routers.

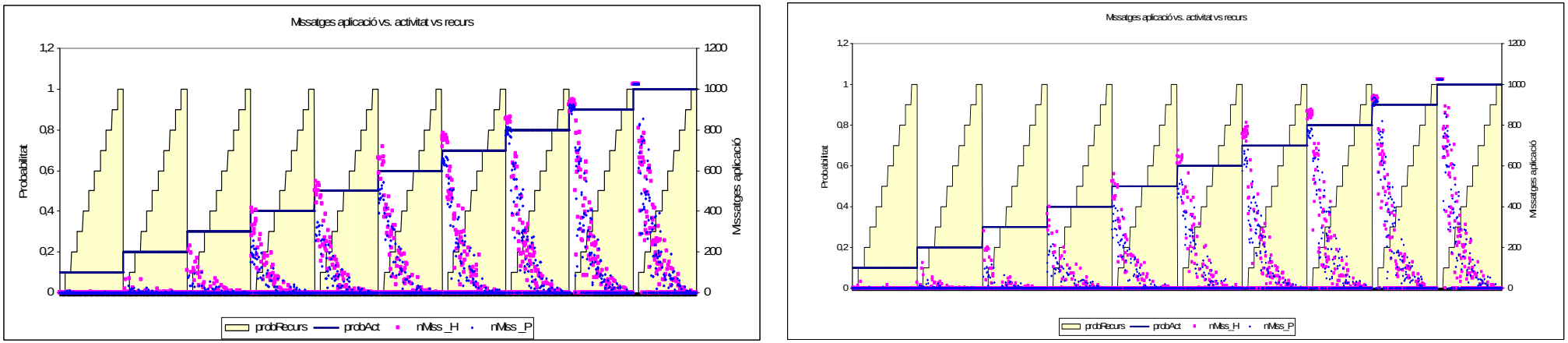


Fig. 8.50. Missatges d'aplicació per hipercub de dimensió 10. 100 i 200 routers.

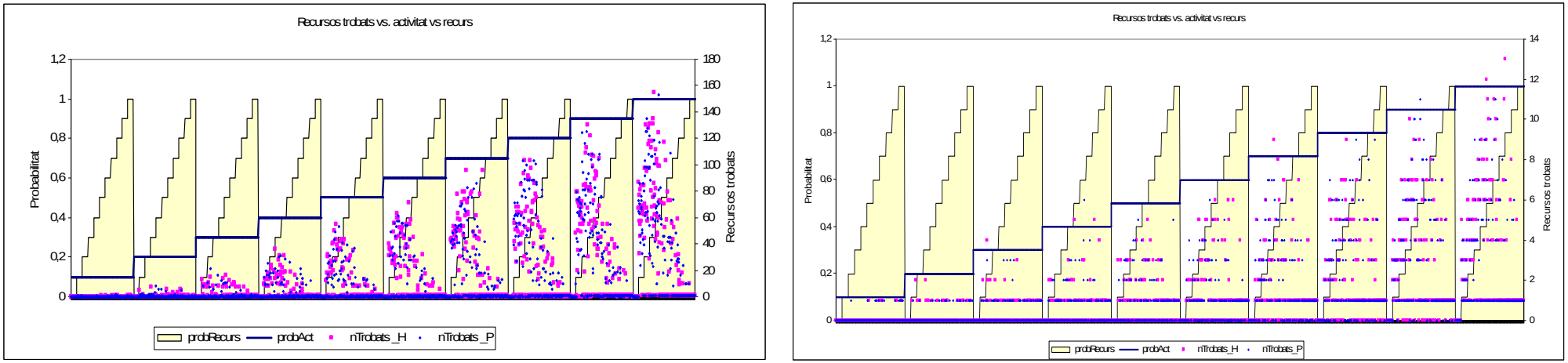
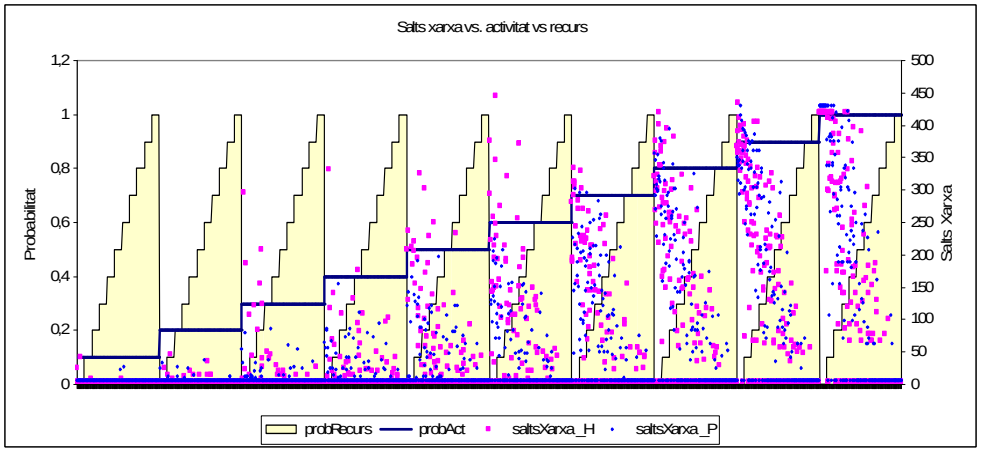


Fig. 8.51. Recursos trobats per hipercub de dimensió 10. 100 i 200 routers.

Les gràfiques següents (de **Fig. 8.52** a **Fig. 8.56**) són anàlegues a les anteriors, sols amb la diferència que en aquest cas, la dimensió de l'hipercub és 5.



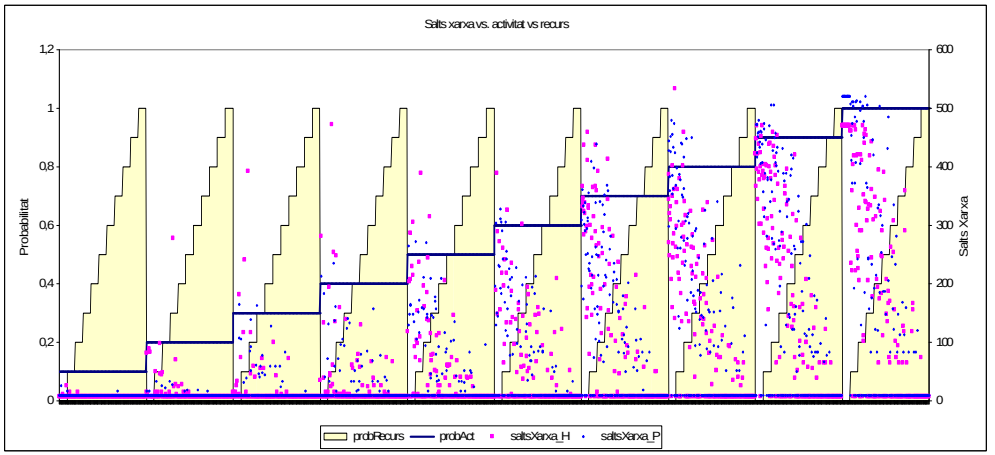


Fig. 8.52. Salts de xarxa per hipercub de dimensió 5. 100 i 200 routers.

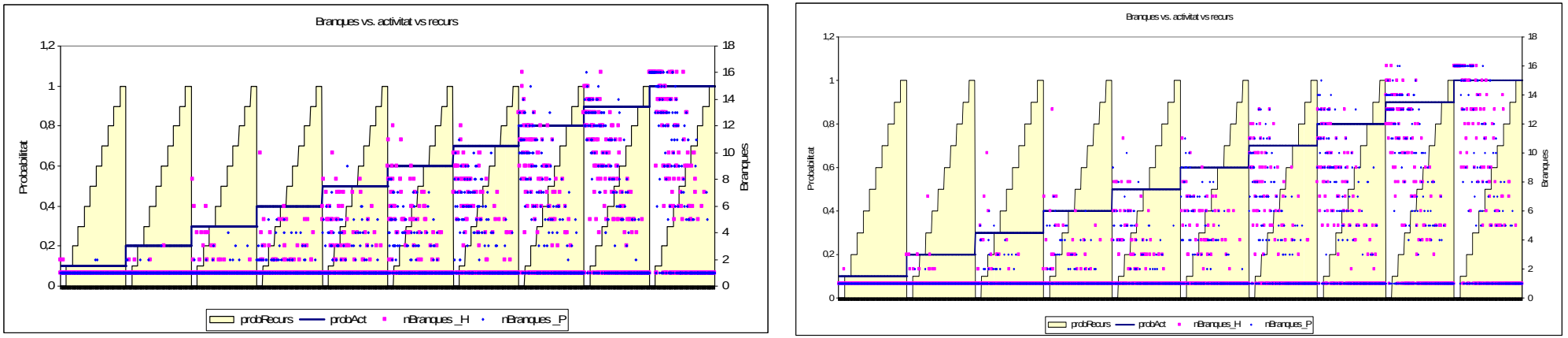


Fig. 8.53. Branques de cerca per hipercub de dimensió 5. 100 i 200 routers.

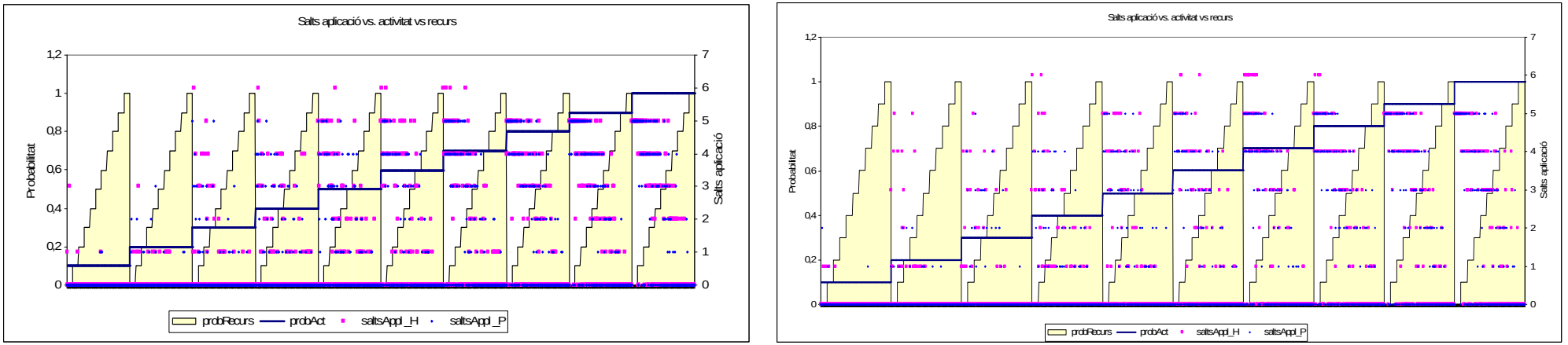


Fig. 8.54. Salts d'aplicació per hipercub de dimensió 5. 100 i 200 routers.

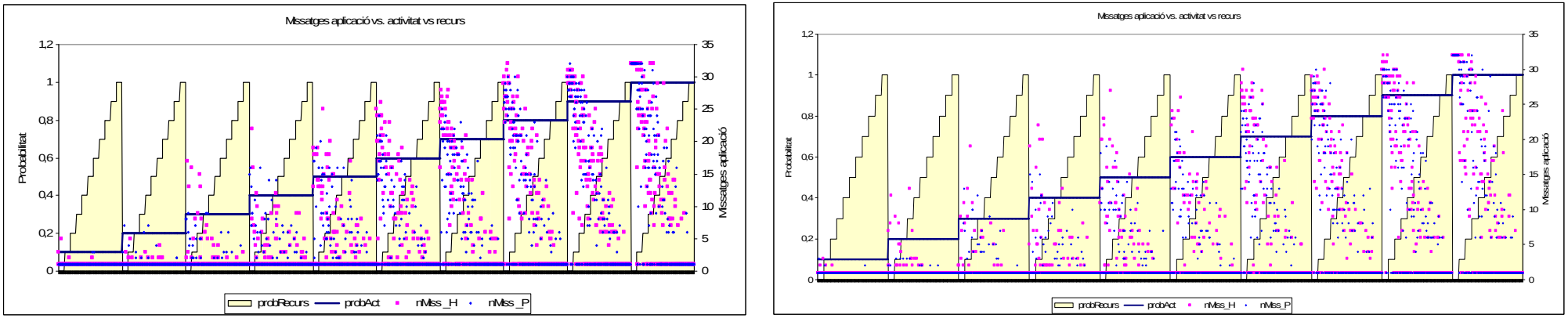


Fig. 8.55. Missatges d'aplicació per hipercub de dimensió 5. 100 i 200 routers.

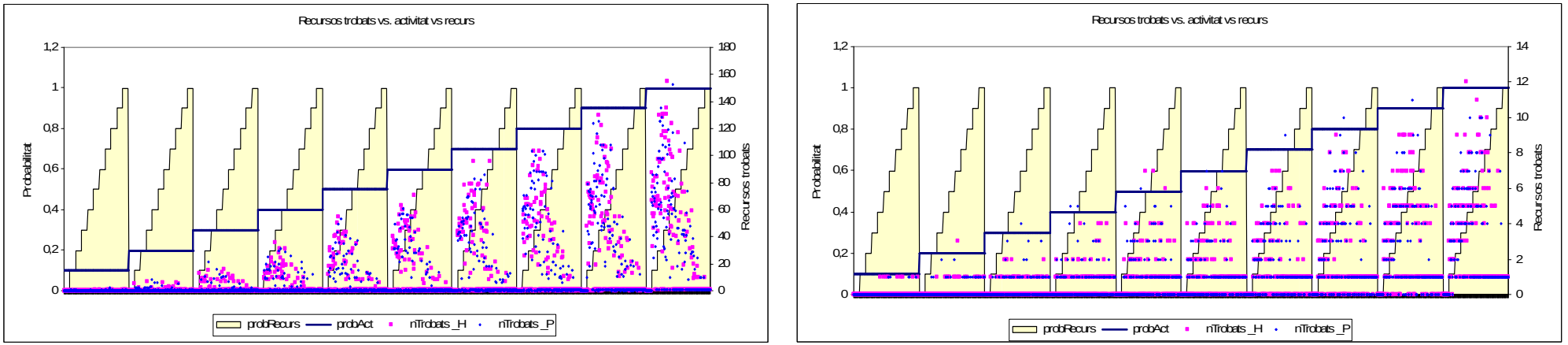


Fig. 8.56. Recursos trobats per hipercub de dimensió 5. 100 i 200 routers.

Una vegada analitzades les dades de les simulacions del primer grup, amb topologia IP estàtica, es pot passar a veure el que es pot extreure del segon grup de simulacions. En aquestes simulacions, la topologia IP varia, mantenint-se estàtica la dimensió de l'hipercub a nivell d'aplicació.

La
Fig. 8.57 mostra la comparació de les mateixes mètriques anteriors per a totes les simulacions realitzades amb hipercub de dimensió 8. El valor de dimensió 8 s'ha triat pel fet de convergir un número important de nodes d'aplicació junt amb una relativa rapidesa amb l'execució de les simulacions. Es pot observar com les mètriques depenents exclusivament del nivell d'aplicació mantenen valors molt constants per a totes les simulacions i per a cada algoritme. Això ve a corroborar la tesi que els algoritmes de cerca són totalment independents de la topologia IP que tenen a sota. En canvi, les mètriques que guarden relació amb el nombre de salts de xarxa si que varien. Es pot veure com en les primeres simulacions, on la topologia està organitzada amb múltiples sistemes autònoms de petites dimensions, es requereixen més salts IP per a cada branca de cerca que no pas en les topologies amb dos sistemes autònoms.

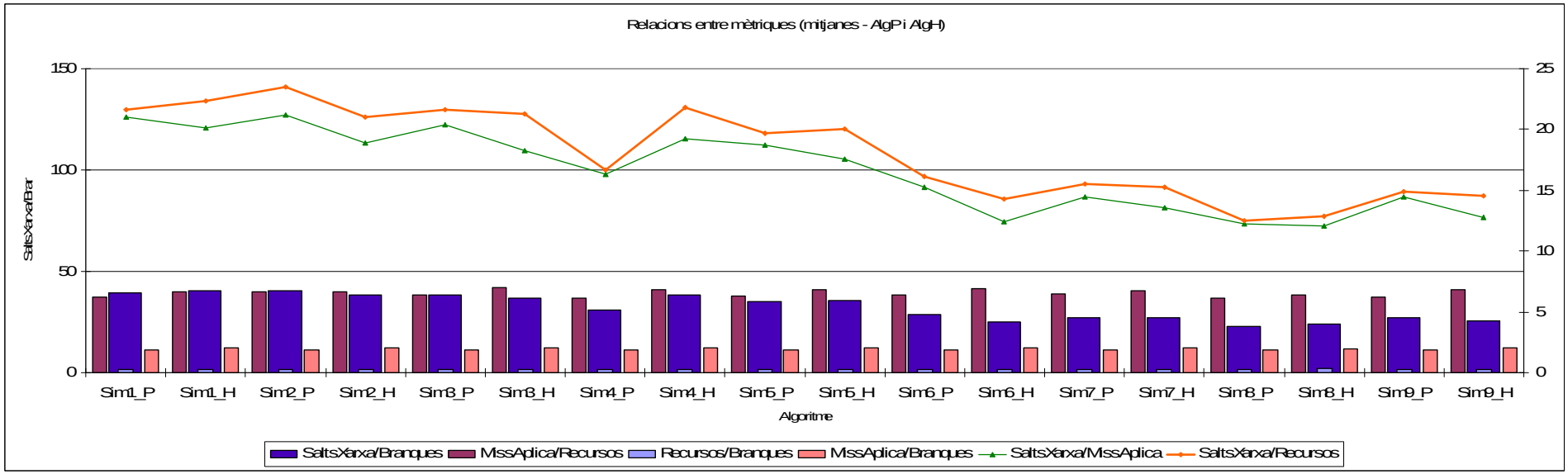


Fig. 8.57. Relacions entre mètriques mitjanes, amb hipercub r=8 i variant la topologia IP.

El bloc de figures que segueix (de **Fig. 8.58** a **Fig. 8.62**) mostra tot el conjunt de valors obtinguts per a dues simulacions del segon grup: Sim1 (gràfiques de l'esquerra) i Sim9 (gràfiques de l'esquerra). (veure **Taula 5.6** per a les respectives configuracions). Com es pot comprovar immediatament, els valors obtinguts per a les dues simulacions segueixen el mateix patró, especialment per a les mètriques de nivell d'aplicació. En el cas dels salts de xarxa, es veure com efectivament en la simulació Sim1, per a les mateixes probabilitats s'acumulen més salts de xarxa que no pas en la simulació Sim9, però tenen distribucions pràctivament iguals.

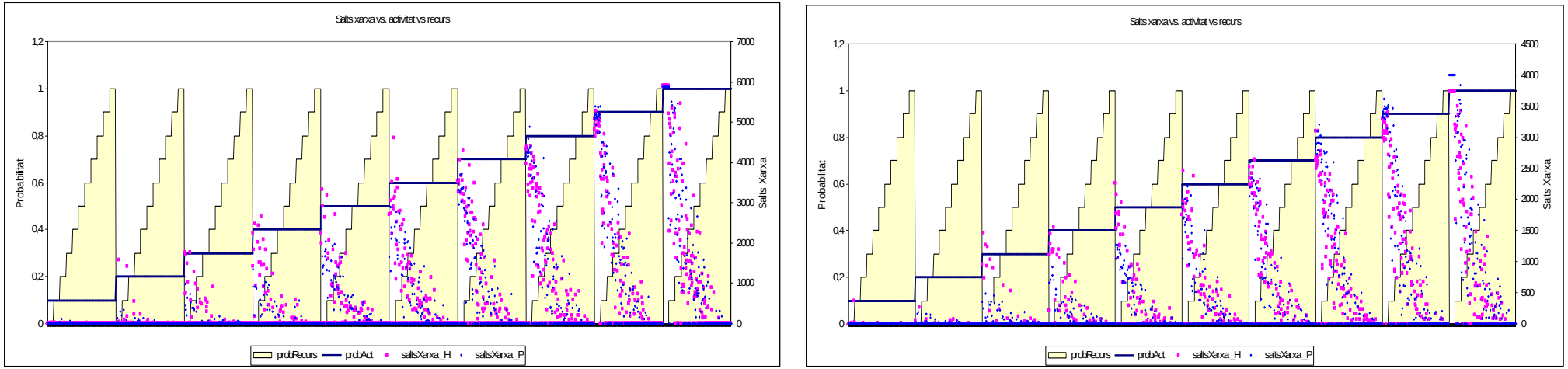


Fig. 8.58. Salts de xarxa en hipercub de r=8. Simulacions Sim1 i Sim9.

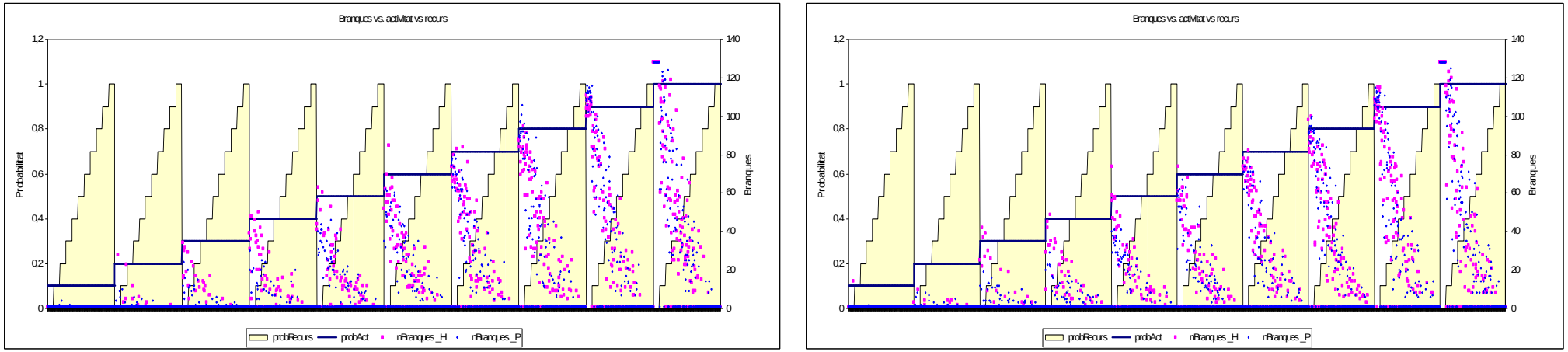


Fig. 8.59. Branques de cerca en hipercub de r=8. Simulacions Sim1 i Sim9.

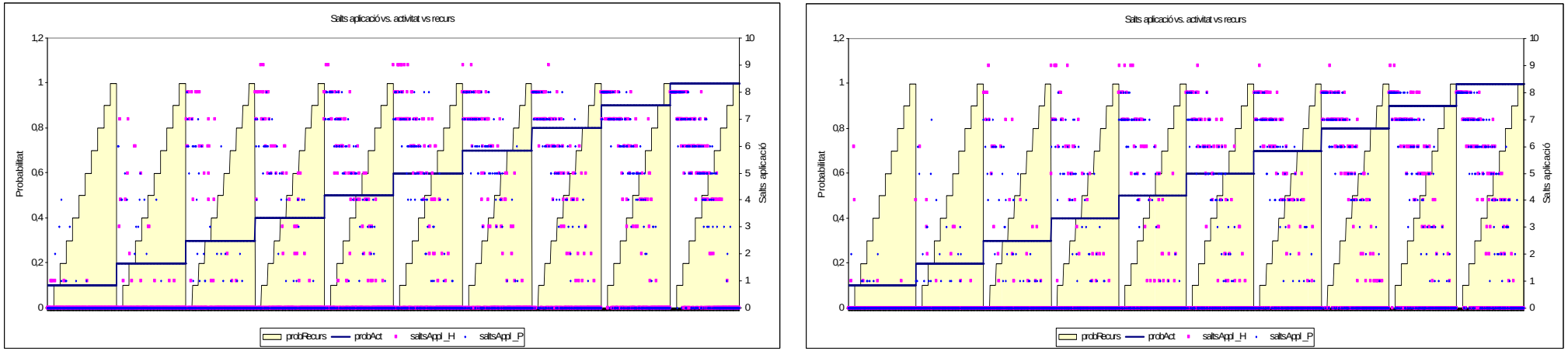
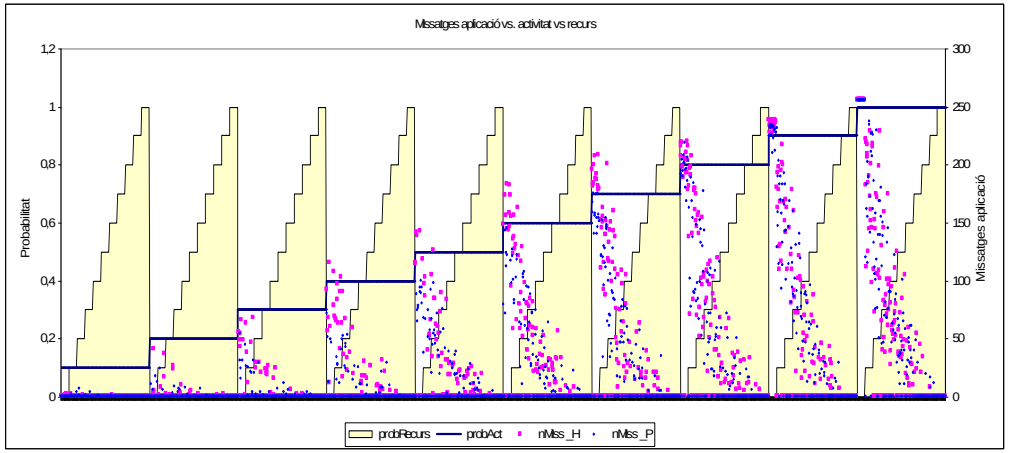


Fig. 8.60. Salts d'aplicació en hipercub de r=8. Simulacions Sim1 i Sim9.



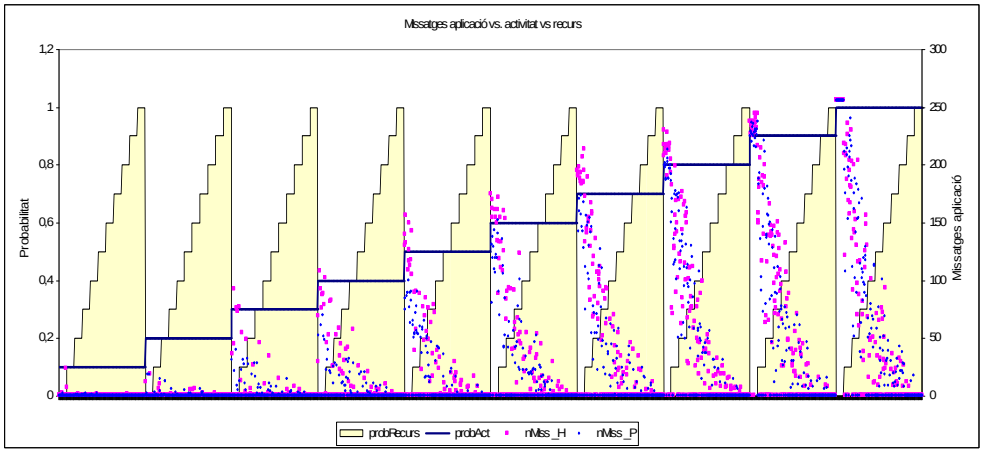
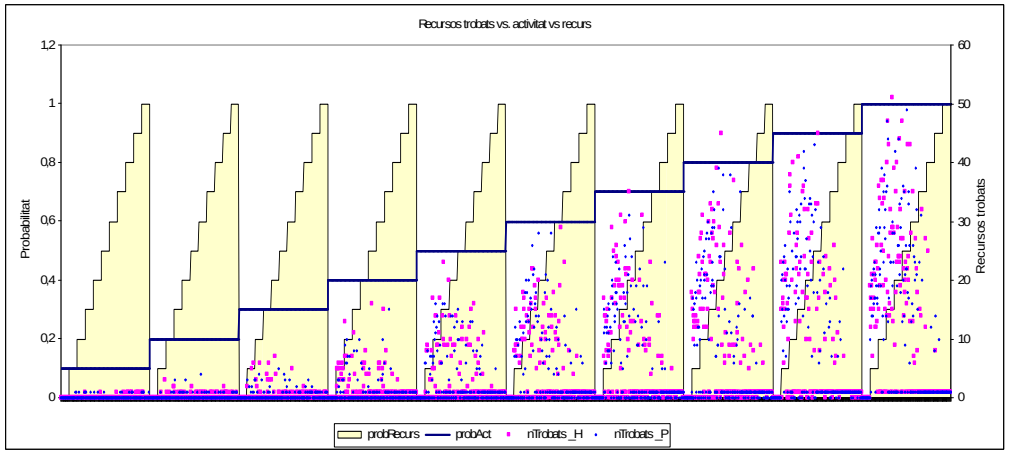


Fig. 8.61. Missatges d'aplicació en hipercub de r=8. Simulacions Sim1 i Sim9.



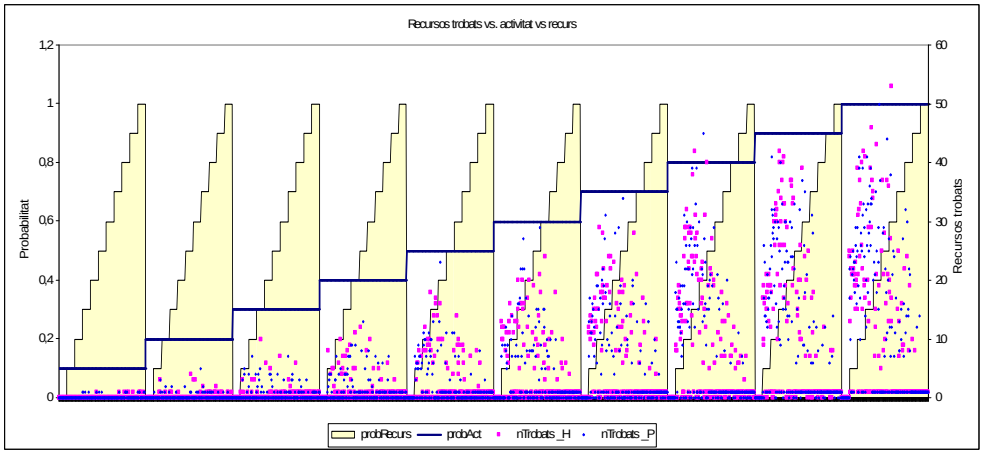


Fig. 8.62. Recursos trobats en hipercub de r=8. Simulacions Sim1 i Sim9.

8.17. Manual de l'Aplicació

Ja que l'aplicació desenvolupada és de tipus consola, no consta d'interfície gràfica i a més llança simulacions que poden requerir l'entrada de paràmetres addicionals per tal d'augmentar la memòria disponible per a la màquina virtual Java, s'ha decidit no empaquetar-la dins d'un arxiu JAR. Així és més còmode llançar les execucions i el procés es controla molt més.

L'execució de l'aplicació es realitza mitjançant la següent comanda:

```
java –Xms256m –Xmx1024m –classpath LIB1:LIB2:. GRID.proves.Main arg0 arg1 arg2 arg3 arg4 arg5 arg6
```

Cada una de les parts de la comanda "java" té el següent significat:

- -**Xms256m**. Configura la memòria inicial de la màquina virtual Java.

- -**Xmx1024m**. Configura el màxim tamany de memòria disponible per a la màquina virtual Java.

- -**classpath** LIB1:LIB2:. . Indica la ruta a les llibreries necessàries per a l'execució de l'aplicació. En aquest cas, a GridSim i BRITE. Les llibreries s'han de separar per ":" i finalitzar la llista amb ".".

- **GRID.proves.Main**. És el programa principal que llença l'aplicació. Tot seguit s'indiquen una llista d'arguements que configuren l'aplicació pròpiament dita.

- **arg0**. Indica el nom de la simulació, el qual apareixerà com a nom dels fitxers de sortida d'estadístiques.

- **arg1**. Indica quin algoritme de cerca ha d'executar. Les dues opcions diponibles són "P" i "H".

- **arg2**. Indica la dimensió de l'hipercub a configurar. Ha de ser un número enter major que 1.

- **arg3**. Indica el tipus d'usuari a crear (veure 4.6.2.). Les opcions són 4:

    - "1". Usuari de cerca complet.

    - "2". Usuari de cerca controlant quins GIS estan en estat no actiu.

    - "3". Usuari que executa un PING contra un element de la xarxa.

    - "4". Usuari que printa taules d'enrutament del router que s'indiqui.

- **arg4**. Indica si la topologia s'exporta en format Otter ("1") o no ("0").

- **arg5**. Indica si es realitza debug de l'aplicació ("1") o no ("0"). El debug representa per la sortida estàndard (línia de comandes) les accions que va duent a terme l'aplicació en cada moment. Cal indica que activar el debug relantitza l'execució de l'aplicació, ja que part de la capacitat del processador es deriva a printar dades per pantalla.

- **arg6**. Indica quin nivell de debug s'ha de representar. Es tenen tres opcions disponibles (si no s'indica cap valor, es fa un debug complet):

    - "0". Debug a nivell de missatges d'aplicació, és a dir, activitat referent a GIS, usuaris i recursos, sense informacions sobre el nivell IP.

    - "1". Debug a nivell IP, és a dir, activitat de routers i enllaços.

    - "2". Debug complet, mostrant tots els missatges de debug possibles.